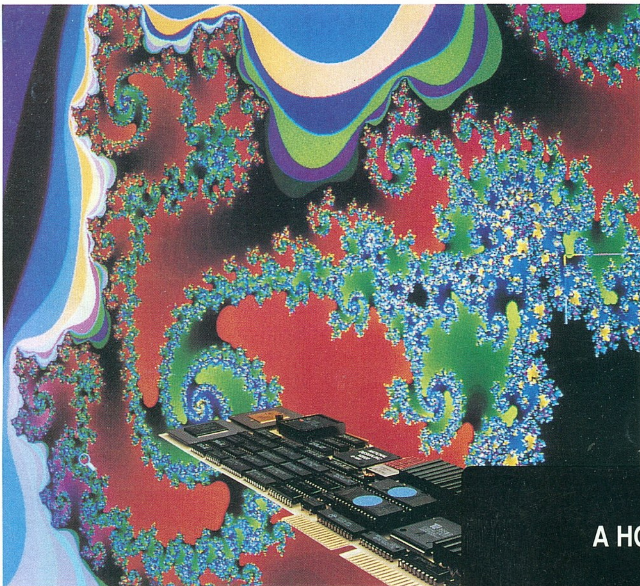
1991 / JÚLIUS

ÁRA: 196 FT

ALAPLAP

MIKROSZÁMÍTÓGÉP MAGAZIN MÁGNESLEMEZ MELLÉKLETTEL



Kód-gettó

TV kontra PC

Metafüggvénytan

Szövegkonvertálás

A HÓNAP TÉMÁJA:

CSOMAGOLUNK

Adatbázis — igényeseknek is

Programozás és a józan paraszti ész

A MÁGNESLEMEZEN:

Nézőke tömörítvényekhez
Az adatsűrítés éLHArcosa
Ablakon be, ajtón ki
Konvertáló kiskapuk
Gyorsolvasás

Adatra vagy objektumra orientálva

A programozási nyelvek világa

Arcmaster fájlmester

Szóval sysop szeretnél lenni?

Közvéleménykutatás

Főnyeremény: 1 db AT
Beküldési határidő:
1991. augusztus 31.

IR Szerviz és Kereskedelmi Kft.



Lehet, hogy vannak nálunk olcsóbbak,
Talán olyanok is, akik jobbat adnak, de az
Biztos, hogy nálunk a minőség megéri az árát!
IR szerviz — a tökéletes összhang

Az Intellrobot szerviz a jövőben önálló cégként,
IR Szerviz és Kereskedelmi Kft. néven dolgozik.
Címünk és telefonszámunk változatlan:
VII. kerület, Kisdiófa utca 6. Telefon/Fax: 121-3230, 141-0880

INFORMÁCIÓKÉRÉS: 09 ▲

ALAPLAP

Mikroszámítógép magazin mágneslemez melléklettel

Megjelenik havonta

Főszerkesztő:
Faklen Pál

Főszerkesztő-helyettes:
Varga János

Szerkesztő:
Jakab Ágnes

Főmunkatárs:
Kis János

A mágneslemez melléklet és a Közkincs szerkesztője:
Verebély Pálné

A Lemezkalauz szerkesztője:
Vékony Tamás

A szerkesztőbizottság tagjai:
Barna László
Boros György
Broczkó Péter
Brüll Károly
Farkas Ernő
Herczeg József
Horváth Imre
Kassay Árpád
Kovács P. Attila
Kónya László
Nagy Gábor
Pintér Gábor
Zoltai Péter

Szerkesztőség, kiadó és hirdetésszervezés:
XI., Karolina út 17.
Budapest 1251
Telefon: 185-1584
Fax: 185-2221

Felelős kiadó:
Sebestyén Ilona igazgató
Cédrus Informatikai Rt.

Nyomdai előkészítés:
Tipoprint Kft., Budapest
Nyomtatás:
Zalai Nyomda, Zalaegerszeg
Felelős vezető: Galla József

Terjeszti a Magyar Posta.
Előfizethető a hírlapkézbesítő postahivatalokonál és a Posta Hírlapelőfizetési és Lapellátási Irodájánál (XIII., Lehel u. 10/a, Budapest 1900), vagy átutalással a 215-96162 pénzforgalmi számra.
Példányonkénti ár: 196 Ft
Évi előfizetési díj: 2 352 Ft

Külföldre terjeszti a Kultúra,
Pf. 149, Budapest 1389

HU ISSN 0865-9788

A HÓNAP TÉMÁJA: CSOMAGOLUNK

(Írta és szerkesztette: Nagy Gábor)

NYÚZÓPRÓBA

SZÖVEGELŐ

SZOFTVERTÉKA

KÖZKINCS

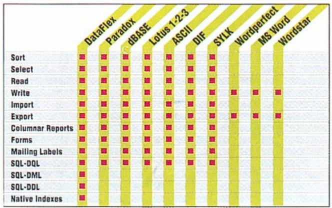
SOLARSOFT KATALÓGUS

GÉPRAJZ

SZERSZÁMOSLÁDA

KILÁTÓ

KIRAKAT



ALAPJÁRAT

FOGÓDZÓ

PROGRAMOZÁSTECHNIKA

VISSZACSATOLÁS

PALETTA

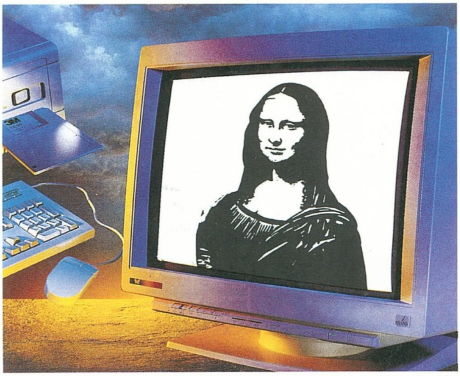
MÁGNESLEMEZ MELLÉKLET ❑

Címlapkép a National Design, Inc. reklámjából

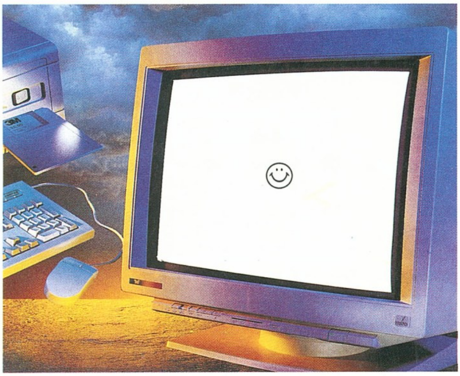
Karikatúrák: Feleki Zoltán

Közvéleménykutató kérdőív

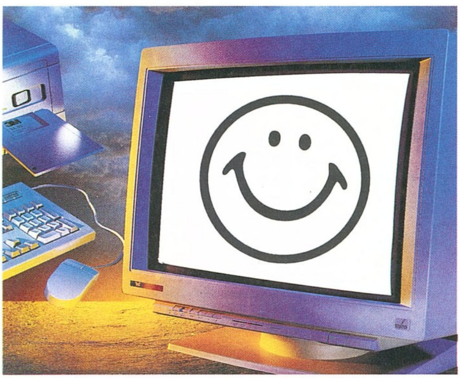
Csomagolunk



Kép beszkennelve



Kép tömörítve



Kép kicsomagolva

Ennél aktuálisabb címet a hónap témájának
nehéz lett volna találni.
Amikor az összeállításon dolgoztunk,
már befejezéséhez közeledett a nagy csomagolás
a szovjet laktanyákban, mi pedig éppen elkezdtük
csomagolni a szerkesztőség
áttekinthetetlen papírgyűjteményét,
hogy átköltözzünk új helyünkre,
a Karolina útra.
Az adattömörítésről és a tömörítőprogramokról
készülő összeállítás munkacíme a
TÖMÖR GYÖNYÖR volt,
de tudtuk — és a saját bőrünkön is éreztük —,
hogy a csomagolás művelete
még akkor sem tömör gyönyör,
ha annak eredménye
valóban örömmel tölthet el bennünket.
Ugyanez elmondható az állományok
be- és kicsomagolásáról is.
Fontos, hasznos műveletek,
de nem nyújtanak közvetlen élvezetet
— ellentétben a számítógépen végzett
számos más munkával...
nem is szólva a játékokról.
Persze sok mindent nem élvezetből,
hanem szükségből csinálunk,
és az adattömörítésre ez nagyon igaz.
„Kiterített" programjaink és adatállományaink
tárolására előbb-utóbb kevésnek bizonyul
a merevlemez kapacitása,
de még floppyval sem győzzük majd.
A tömörítésnek és a tömörített állományok
kibontásának mindennapi rutinművelette kell válnia.
Láttuk azonban azt is,
hogy ma még a számítógépet általában
nagy biztonsággal használók közül is
milyen sokan bizonytalanodnak el,
ha tömörítőprogramokat kell használniuk,
tehát szükség lenne valami „útikalauzra".
Mostani összeállításunk
az első lépés ebben az irányban.
Másodikként az Alaplap Könyvek sorozatban
ugyanezt a témát hamarosan részletesebben
is feldolgozzuk.
Most viszont csomagoljunk!
Be és ki.

Helyhiánnyal küzdők mentőöve

Útikalauz a tömörítéshez

A hazai PC-felhasználók körében tipikus helyzetnek tekinthető, hogy szűkös anyagi lehetőségek korlátai között kell egyrészt nagyobb tárolókapacitásra szert tenni, másrészt az állományokat minél kisebbre tömörítve tárolni, anélkül, hogy a programok használatában zavar keletkezne. Ennek a nem könnyű feladatnak a megoldásához jó volna tudni, hogy egyáltalán milyen lehetőségek közül választhatunk. Az alábbi írás ehhez igyekszik egy kis áttekintést adni.

A „jó öreg" Commodore, Spectrum és egyéb 8 bites gépeken eleinte az egyetlen elérhető adat- és programrögzítési eszköz a kazettás magnó volt, lassú és megbízhatatlan szalagos adathordozóval.

A floppy már rendkívül jól használható, de költséges és hamarosan elborítja a polcokat és az asztalt. Az ötvenedik megtelt lemezen túl nem is tudtam fejben tartani, melyiken mi van. Nem hiszem, hogy bennem volt a hiba, mert barátaim hasonló cipőben járnak.

Amikor áttértem a winchester használatára — természetesen továbbra is jól megőrizve floppyn a backupokat — a 30 megás merevlemez szinte végtelenül nagynak tűnt... Azután igen hamar szűknek bizonyult, és elkezdődött a selejtezések, az átrendezések időszaka. Végül eljutottam oda, hogy bár minden ritkán használt állományomat floppyra raktam, a merevlemez mégis állandóan csordultig volt. Új gépet, új rendszert, új winchestert kellett volna beszerezni... Ilyen beruházásokra azonban nem futotta. Akkor hogyan és merre tovább?

COM és EXE fájlok tömörítése

Az utóbbi hónapokban különféle új winchestervezérlő kártyákat hirdetnek, amelyek megduplázzák a kapacitást, de mivel ezek fizikailag is többet írnak a lemezre, hamarabb és gyakrabban jelentkezhetnek a lemezhibák. A különféle backup programok (PC-Backup, Norton Backup) csak kiegészítésként jöhetnek szóba, hiszen nemcsak őrizgetni akarok, hanem munkaállományaimat nap mint nap használni és lemezen szállítani is. Nézzünk hát körül, hogy egyáltalán milyen alkalmas módszerek kínálkoznak!

E programcsalád képviselői a végrehajtható COM és EXE fájlokat tudják elég jó hatásfokkal és funkcióvesztés nélkül összezsugorítani. A megoldás azonban csak átmeneti, és kiegészítő eljárásokat, ellenőrzéseket is igényel, mert nem mindegyik program tűri jól a zsugorítást, néha éppen a legszellősebb szerkezetű állományok maradnak érintetlenek. Akkor célszerű alkalmazni, ha a kiválasztott programokat lemezről vagy lassú winchesterről akarjuk futtatni. Ilyenkor ugyanis a beolvasás jelentősen gyorsul, a kibontással járó lassulás pedig elhanyagolható.

Két ilyen megvizsgált rendszer közül a PKLITE kezelése egyszerű, mind a COM, mind az EXE fájlokat tömöríti. Nem lehet azonban túl kicsi vagy belső overlay-technikát alkalmazó programokat összenyomni vele. Ezek nagy részét maga a tömörítő jelzi. Előnye még, hogy a tömörített programok eredeti állapota szinte mindig visszaállítható.

A tömörítés — több nézőpontból

I. A tömörítő típusa szerint

1. Backup programok (PCTools/C.P.Backup, NBACKUP).
2. COM és EXE fájltömörítők (PKLITE, LZEXE).
3. Fájltömörítők (LHA, PAK, PKZIP, ARC, ZOO, LHARC, PKARC, PKPAK).
4. KK tömörítő (KK/KXK/KKINS/KKUNINS).
5. Winchesterbővítők (STACKER, NEWSPACE).

II. Adattípus szerint

1. Szöveges adatállományok (Főleg TXT, DOC fájlok).
2. Grafikus adatállományok (Pixel- és vektorgrafika).
3. Táblázat- és adatbázis-állományok (DBF, WK?).
4. Programállományok (COM, EXE, OV?, SYS).
5. Előzőleg már tömörített archív állományok (ZIP, ARC, LZH).

III. Felhasználók szerint

1. Teljesen kezdő — „beesett" az utcáról. (STACKER)
2. Kezdő — már látott számítógépet. (PKLITE, LZEXE, PKZIP)
3. Haladó — angolul tud, a DOS-t ismeri. (LHA, PAK, PKZIP)
4. Profi — mindent tud vagy tudni akar. (ZOO, PAK, LHA, PKZIP, PKPAK, ARC, LZEXE)
5. Programozó — mindent jobban tud. (Írjon magának egyet!)

IV. Rendeltetés szerint

1. Biztonsági (backup-) másolat (PC-BACKUP, NBACKUP, ZOO).
2. Program- és adattovábbítás (PAK, LHA, PKZIP).
3. Munkaterület bővítése (STACKER, NEWSPACE, KK).
4. Régebbi verziók megőrzése (ZOO).

A fájltömörítés művelete

A fájltömörítők a kiválasztott állományokat (ha kérjük) a könyvtárszerkezettel együtt becsomagolják egy tömörített archív állományba. Közben törölhetünk, áthelyezhetünk, aktualizálhatunk, jelszóval titkosíthatunk, kommentezhetünk fájlokat. A kész archív fájlnak van egy jól használható tartalomjegyzéke (archív katalógus), amely nagy segítség az adatállományok és programok nyilvántartásban.

Az így előállított, tömörített archív állományokkal szinte azt csinálunk, amit akarunk. Ki lehet csomagolni egyszerre mindent, és külön-külön a kiválasztott állományokat. Lehet bővíteni, aktualizálni, belekukkantani, törölni, nyomtatni, ellenőrizni, önkicsomagolóvá (SFX) alakítani stb.

A fájltömörítők előnye a jó tömörítési hatásfok, és hogy a létrehozott archív állományok könnyebben utaztathatók lemezeken és különféle más kommunikációs csatornákon.

A fájltömörítők hátránya, hogy aktualizáláskor vagy használat előtt mindig ki kell bontani az archívba tömörített állományokat, és ha csak átmenetileg is, de ilyenkor dupla helyet foglalnak el a lemezen. Visszaüt a tömörítőprogramok sokfélesége is, mert néha nehéz vagy hosszadalmas a különféle archív formátumok között konvertálni az adatokat, s könnyen előfordul, hogy a szükséges tömörítő/kicsomagoló/konvertáló segédprogramot nem ismerjük, vagy nincs elérhető helyen.

Szerencsére a sokféleségnek előnyei is vannak. Mostanra már a különféle tömörítők parancskészlete és szintaxisa kezd egy íratlan „szabvány" felé közeledni, s aki egyetlen tömörítő használatát is legalább elemi szinten elsajátította, az az összes többivel pillanatok alatt megbarátkozik.

A szabványosodás egyik jellegzetes eredménye, hogy csak az a tömörítőprogram lehet piacképes, amelyik legalább az alábbi funkciókkal rendelkezik:

Add = bepakolás tömörítéssel.
Freshen = frissítés, újat nem visz be.
Update = aktualizálás, újat is bevisz.
Move = áthelyezés, a forrás törölve.
Delete = fájltörlés az archívból.
Extract = kibontás, visszaállítás.
Test = az archív épségének tesztelése.
View List = az archívkatalógus kiírása.
Console = képernyőn való megjelenítés.
Print = kinyomtatás.

Emellett biztosítania kell a programnak valamiféle megjegyzés hozzáfűzésének lehetőségét és önkibontó (SFX) archív létrehozását. Ez tehát a minimum, amit egy programnak tudnia kell.

A fájltömörítők többségének parancsai az imént felsorolt angol szavak kiemelten jelzett betűi és maguk a szavak.

Tömörítők tesztelve

A vizsgált fájltömörítőket önkényesen és teljesen véletlenszerűen kiválasztott állományokon teszteltem.
A szövegállományokhoz saját kézirataim közül választottam ki hetet.
A grafikák tesztjéhez a Genius DRGenius programmal készítettem három PIC fájlt.
A táblázatok tömörítésének alanyául a Lotus Learning System 74 K-s állományát szemeltem ki.
Az adatbázis-állományok tesztjéhez saját dBase állományaim közül választottam ki az első tízet.
A bináris állományok tesztjéhez gépem DOS könyvtárának 22 db EXE és COM állományát használtam.

Szövegek (7 .DOC állomány, 58 250 bájt)

LHA 2.10	26514	1.	45,5%
PAK 2.50	27328	2.	46,9%
PKZIP 1.10	28096	3.	48,2%
KK	31796	4.	54,6%
PKARC 3.60/ PKPAK 3.61	32202	5.	55,3%
ZOO 2.01	32483	6.	55,8%
ARC 6.02	33262	7.	57,1%

Grafikák (3 .PIC fájl, 19 456 bájt)

LHA 2.10	9575	1.	49,0%
PAK 2.50	10529	2.	53,9%
PKZIP 1.10	10619	3.	54,3%
ZOO 2.01	11780	4.	60,3%
PKARC 3.60/ PKPAK 3.61	11782	5.	60,3%
ARC 6.02	11944	6.	61,1%

Táblázatok (Eredeti .WK1 fájl, 74 118 bájt)

LHA 2.10	50337	1.	67,9 %
PAK 2.50	51372	2.	69,3 %
PKZIP 1.10	54208	3.	73,1 %
ZOO 2.01	69317	4.	93,5 %
PKARC 3.60/ PKPAK 3.61	69513	5.	93,8 %
ARC 6.02	71116	6.	95,9 %

Adatbázisok (10 .DBF állomány, 384 496 bájt)

LHA 2.10	42141	1.	11,0 %
PAK 2.50	53139	2.	13,8 %
PKZIP 1.10	54644	3.	14,2 %
PKARC 3.60/ PKPAK 3.61	82620	4.	21,5 %
ZOO 2.01	83485	5.	21,7 %
ARC 6.02	87044	6.	22,6 %

Programok (22 DOS állomány, 242 693 bájt)

LHA 2.10	148197	1.	61,1 %
PKZIP 1.10	153907	2.	63,4 %
PAK 2.50	160503	3.	66,1 %
PKLITE	163415	4.	67,3 %
PKARC 3.60/ PKPAK 3.61	183892	5.	75,8 %
ARC 6.02	191636	6.	79,0 %
ZOO 2.01	192778	7.	79,4 %
LZEXE 0.91	210623	8.	86,8 %

Összesített sűrítési rangsor

1. LHA
2. PAK
3. PKZIP
4. PKARC/PKPAK
5. ZOO
6. ARC

Néhány elterjedt program

A legkisebb méretű, s egyben a legnagyobb tömörítést elérő program jelen pillanatban az LHA.EXE. Nem szupergyors, de tömörítőképessége igen értékes tulajdonság.

A NoGate cég PAK programja meglehetősen terjedelmes, bár sokat is tud. Közvetlenül az LHA után következik tömörítésben, adatkonverziós lehetőségei pedig egyedülállóak. Parancsait mindenki saját igényeihez illesztheti. Állományokat áthelyezni (move) nemcsak befelé, hanem kifelé is lehet vele, vagyis az éppen kipakolt állományokat egyetlen paranccsal egyben törölni is tudjuk az archívból.

A PKZIP egyszerre nyújtja a gyorsaságot, a nagyfokú tömörítést és az adatbiztonságot. Extra szolgáltatásai, főleg a nyomtatás és a sérült állományok felélesztése területén különlegesen rugalmassá teszik. (Kinyomtatásban egyszerűen verhetetlen.)

A PKPAK a régi PKARC utóda. Egy kicsit kezd elavulni, de használják, mert még mindig elég sok ARC tömörítésű állomány forog közkézen. Alapprogramnak számít. Szolgáltatásai néhány kivételtől eltekintve megegyeznek a PKZIP csomagéival.

A ZOO három tulajdonsága miatt érdekes: egészen hosszú (maximum 64 K) megjegyzéseket is elfogad, állománygenerációkat tárol egymás mellett (fejlesztők, figyelem!), és mellékelték hozzá a program forráskódját is.

A tiszteletreméltó korú ARC minden tesztben hátul végzett. Egyetlen erénye, hogy archívból tudja futtatni az általa bepakolt EXE és COM állományokat.

Egy köztes kategória

Masa István KK / KXK / KKINS / KKUNINS programcsomagja azért érdemel külön figyelmet, mert átmenetet képez a fájltömörítők és a winchesterduplázók között. Lehetővé teszi, hogy a becsomagolt állományokat szekvenciálisan kibontva újra olvassuk. Igen jól használható olyan szövegek archiválásához, amelyeket nem akarunk módosítani, csak olvasni vagy nyomtatni, illetve más programokba átküldeni vagy másolni. Lehetséges alkalmazási területei a szövegeket szekvenciálisan felhasználó help-rendszerek, programismertetők, valamint egyes Hypertext-rendszerek.

A szövegállományokat a csomag KK.EXE programjával lehet összezsugorítani. Az így tömörített állományokat az Alaplap mágneslemez-mellékletén is használt rezidens KKINS program segítségével mindenféle paraméterezés nélkül kibontva láthatjuk mindaddig, amíg a KKUNINS programmal be nem csukjuk. Közben tetszőleges programmal lehet olvasni, másolni stb. A végleges kibontás kétféle is lehet: a KXK programmal, vagy úgy, hogy a KKINS meghívása után, de még a KKUNINS használata előtt a programot más néven egyszerűen bemásoljuk egy tetszőleges könyvtárba.

A program szinte pofonegyszerűen használható, s hihetetlenül kevés helyet foglal. Alkalmazása 1 kilobájtnál nagyobb állományok tömörítésekor célszerű, mert ennél kisebb szövegek esetén még méretnövekedést tapasztalhatunk.

Winchesterduplázók

Ennek a programcsaládnak két tagja is felbukkant Magyarországon. A Stacker kereskedelmi program két lemezzel (5,25" és 3,5"), kézikönyvvel, koprocesszoros és szoftveres változatban kapható. A másik, a NewSpace egy shareware program (Solarsoft #475), amit regisztráltatni lehet.

A NewSpace azoknak ajánlható, akiknek csak egy 20 megabájtos winchesterük van és a közeljövőben nem akarják a gépet új vagy nagyobb merevlemezzel bővíteni, mivel a program egy gépen csak egyszer installálható. A 499 forintos ár mindazoknak igen kecsegtető, akik nem tudnak ezreseket áldozni gépük bővítésére. Mielőtt mindenki rohanna beszerezni a programot, fel kell hívnom a figyelmet, hogy multitasking rendszerekkel (Windows, DESQview) nem tud együttműködni, és a hálózatokban csak a terminálokon működtethető, a szervergépen nem. Talán a következő verziókban ezeket a problémákat is megoldják. A program nem tömöríti a COM és az EXE állományokat, az AUTOEXEC.BAT-ot, a SYS-t és a többi device drivert, valamint a rejtett állományokat és a rendszerállományokat. A lemezen egy 38 oldalas leírás is található.

A Stacker koprocesszoros változata nem tartozik az olcsó programok közé, de megéri az árát. Az installálás után jó darabig nem lesz helyproblémánk. Az átlagfelhasználónak viszont elegendő a szoftveres, olcsóbb változat is (20 ezer Ft körüli ár). Egy gépen tetszőleges számú merevlemezre, illetve partícióra installálható. Az egyetlen hátrány, hogy partíciónként a 30 K mellett további 3—3 K RAM-ot vesz igénybe. DR-DOS 5.0-val, vagy memóriamenedzser-programokkal azonban a 640 K fölötti, illetve az EMS-területre is telepíthető.

Mindenkinek a megfelelőt

Teljesen kezdő felhasználó számára ideális megoldás a Stacker vagy a NewSpace használata. Az installálás után ugyanis egyáltalán nem kell azzal törődnie, hogy a tömörítés rendben menjen, ez a program dolga. El is végzi, szinte észrevétlenül.

Aki nem teljesen kezdő, és gyakrabban kell programjait és adatait lemezre mentenie, annak a winchesterduplázók mellett a különböző fájltömörítők használatát ajánlom. Ha a sebesség fontos, akkor a PKLITE, LZEXE, PKZIP vagy a PAK az optimális megoldás, ha a tömörség, akkor az LHA, a PKZIP vagy a PAK. Ha az extra szolgáltatások lényegesek, akkor először végig kell nézni, melyik program mit kínál.

Mekkorák a tömörítők?

PKLITE 1.03	12 998 bájt
KK	14 361 bájt
LHA 2.10	33 951 bájt
PKPAK 3.61	36 832 bájt
PKARC 3.6	36 896 bájt
ZOO 2.01	40 078 bájt
ARC 6.02	84 170 bájt
PKZIP 1.10	89 236 bájt
PAK 2.50	106 060 bájt

Haladóknak az LHA, a PKZIP és a PAK programokat mindenképpen ajánlom, ha még nem használnák. A többit úgyis megnézik, megszerzik magánszorgalomból.

Profiknak, programozóknak legfeljebb arra hívom fel a figyelmüket, hogy a fájltömörítőknek is vannak olyan szolgáltatásaik, amelyeket általában csak a kézikönyv sokadszori fellapozásakor fedezünk fel.

Aki pedig maga akarja megírni saját tömörítőjét, nem árt, ha belekukkant a ZOO 2.01 csomagba, melyben a forráskódok is megtalálhatók, méghozzá különféle operációs rendszerekre kihegyezve.

Mikor melyiket

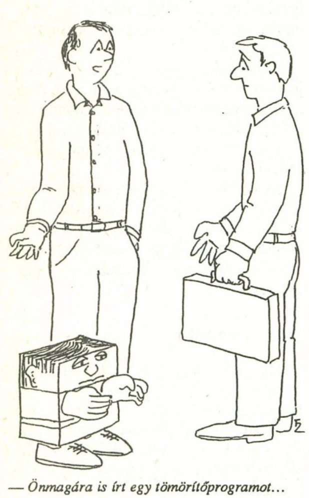
Ha egyszerűen csak egy biztonsági másolatot akarunk készíteni winchesterünk tartalmáról, használjunk backup-programokat (pl. Norton Backup vagy PCTools, C.P. Backup). A fájltömörítők könyvtárszerkezet-elmentő képessége is segítségünkre lehet, ha egész könyv-

tárakat akarunk egyetlen archívba tömörítve tárolni.

Amikor a programok és adatok továbbításának vagy cseréjének a meggyorsítása a cél, egyértelmű a fájltömörítők előnye a többi típussal szemben. Az összetartozó program- és adatállományokat egyetlen vagy csak néhány archív (vagy SFX archív) fájlba becsomagolva, könnyebben kezelhetők állományaink. A kommentezési lehetőségek módot adnak arra, hogy használhatóbb megjegyzésekkel lássuk el a legfontosabb tömörített állományokat. A jelszavas titkosítás lehetősége sem utolsó dolog. A legelterjedtebb programok: PKZIP, PAK, PKPAK, LHA(RC), ZOO, ARC, illetve az EXE és COM állományok tömörítői közül a PKLITE és az LZEXE csomag.

Amikor csak a merevlemez rendelkezésre álló szabad felületét akarjuk növelni úgy, hogy az installálás után többé ne kelljen paraméterezéssel, ki- és becsomagolással foglalkoznunk, használjunk winchesterduplázó programot (Stacker, NewSpace). Szövegállományainkat flopp-lemezeinken, amelyekkel a Stacker nem foglalkozik, a KK-val vagy a fájltömörítők egyikével tömörítve érdemes tárolni.

Ha egy szöveget vagy más adatállományt úgy akarunk megőrizni, hogy azt később közvetlenül ne módosíthassák, s csak a kiolvasást akarjuk támogatni, jó szolgálatot tehet a KK csomag. Ha fontos a régebbi verziók megőrzése, hasznos lehet a ZOO program állománygenerációkat kezelő képessége.



— Önmagára is írt egy tömörítőprogramot...

Archivált állományok konverziói

Oda – vissza, össze – szét

A fájltömörítők alkalmazásának egyik legnagyobb hátránya, hogy nincs kötelezően használt szabvány az állományformátumokra. A tömörítőprogramok zöme — tisztelet a kevés kivételnek — csak saját formátumát fogadja el, és csak saját tömörítő algoritmusait használja. Ha kapcsolatba lépnek egymással, állandóan „tolmácsra" van szükség.

A gond ott kezdődik, ha több helyről kapunk különböző tömörített archív állományokat, s ezeket akarjuk egyszerűen kipakolható, közös formátumra hozni. Leggyakrabban először át kell konvertálnunk őket. Olyankor is szükség van erre, ha régebbi verzióval tömörített állományainkat a legújabb tömörítő verzióval még kisebbre akarjuk összenyomni. Néhány tömörítőnek azonban már erre is van külön parancsa vagy segédprogramja.

Az archív konverziónak két alapvető módszere van, a közvetett és a közvetlen. A közvetett konverzió lényege, hogy a kibontó és becsomagoló program (vagy egy átpakoló segédprogram) segítségével első lépésben kibontjuk a forrásarchívot egy átmeneti könyvtárba, második lépésként besűrítjük az új archívba, végül pedig az átmeneti könyvtárból töröljük az ideiglenesen kibontott állományokat. A közvetlen konverziót használó programok gyorsabbak, mert konvertálás után a forrásarchívból közvetlenül az új archívba rámolnak át. Átmenetileg azonban a forrásarchívból éppen feldolgozott fájl négy példányban van a lemezeken (forrás, átmeneti munkafájl, új archív, régi archív). Ezzel az átmeneti sokszorozódással a közvetett konverziónál is számolni kell, amikor több forrást viszünk át egyetlen célállományba. (Nem árt az óvatosság, ha winchesterünk közel van a telítődéshez.)

Bármelyik módszert használjuk is, előbb mérlegeljük, hogy miért éppen az adott tömörítőt választották az archiváláshoz. Nem árt azt is megnézni, (vagy megkérdezni), milyen kísérő információkkal készítették el a forrásfájl állományait.

Saját tömörítőnk kiválasztását ne bízzuk a véletlenre! Ha állandóan ugyanazzal a tömörítővel dolgozunk, legalább önmagunkkal mindig kompatibilisek maradunk. De az is helyes elv, ha minden esetben az adott feladathoz optimális programot választjuk. Figyeljünk oda arra is, hogy kik használják majd az új archív állományokat. A felhasználó számára még ismeretlen tömörítőt illik egy leírás kíséretében mellékelni a tömörített állományhoz.

A fontosabb konvertáló programok

— REZIP.EXE — A PKUNZIP és a PKZIP segítségével alakít át régi verzióval készült ZIP állományokat új ZIP állományokká. Frissítési célokra is alkalmazható.

— TOZIP.EXE — A PKUNPAK és a PKZIP segítségével konvertál .ARC állományokat .ZIP állományokká. Csak egyirányú konverziót tesz lehetővé.

— 2LZH.EXE — ARC, PAK és ZIP állományokat konvertál LZH formátumúakra a PKXARC, PAK, PKZIP és az LHARC segítségével. Ez is csak egyirányú konverziót biztosít.

— LHDIR — Egyéb szolgáltatások mellett az LHARC SFX-ből csinál LZH fájlt. Ilyenkor az SFX fejet szedi le.

— SHEZ 5.9 — A SHEZ a megfelelő kibontó- és tömörítőprogramok segítségével az elterjedtebb archívok bármelyikéről bármilyen más archív formátumra tud konvertálni. A legtöbb tömörítővel SFX állományokat is létre tud hozni, valamint azok egy részét vissza is tudja alakítani.

— PAK 2.50 — Sokoldalú szolgáltatásokat nyújt. Oda-vissza konvertál az alábbi archívtípusok között: PAK (ez az alapértelmezése), ARC, ZIP, valamint az ezekből képzett SFX-típusú EXE. Ráadásul a BBS-ekben előforduló SDN állományokat is saját formátumára tudja átalakítani, a szükséges ellenőrzéseket is elvégezve.

Az „SFX-automata"

Önkicsomagoló tömörítvények

A tömörítőprogramok használatának van néhány árnyoldala is. Adódnak például olyan helyzetek, hogy sem megnézni, sem kibontani nem tudunk egy tömörített állományt, mert a megfelelő kicsomagoló állomány nincs kéznél, vagy nincs az elérési útvonalon, esetleg nem ismerjük a kibontóprogram helyes használatát. Ezeket a problémákat is kiküszöbölik a sokkal kényelmesebben használható, önkicsomagoló tömörített állományok, amelyek a „mindig rezidens" Enter leütésére önként kiterítik teljes tartalmukat (self extract). Szakmai becenevük: SFX.

Az SFX-állományok különleges természetét célszerű valahogy jelezni. Vannak, akik a fájlnévbe illesztenek be egy jelölő karaktert (pl. #), mások emellett rövid szöveges állományt is tesznek mellé (readme), amiben leírják, mi van az SFX-ben és hogyan ajánlatos kinyitni. Az igazán figyelmesek pedig külön indító .BAT állománnyal szerelik fel.

Milyenek az SFX-ek?

— Csak 1—14 kilobájttal nagyobbak az eredeti archívnál.

— Kinyitásukhoz nem kell más, csak a céllemezen elegendő üres terület.

— Többnyire ugyanúgy, vagy majdnem ugyanúgy kezelhetők, mint a hagyományos archívok.

— Egyes programcsomagok SFX-állományai extra szolgáltatásokkal is rendelkeznek.

— Tesztelni lehet épségüket.

Hogyan történik az SFX-állományok előállítása az egyes tömörítőprogramokkal, és mi kell hozzá? Ennek részletes ismertetéséről itt terjedelmi okokból le kell mondanunk, csak egy példát említünk, de az Alaplap Könyvek sorozatban megjelenő könyv ezeket az információkat a többi tömörítőprogramra vonatkozóan is tartalmazza. Pl. a PKZIP 1.02 szükséges állományai:

— MAKESFX.COM (a PKZ102.EXE tartalmazza).

— PKSFX.PRG (a MAKESFX hozza létre a PKZ102.EXE állományból).

— ZIP2EXE.EXE (a PKZ102.EXE tartalmazza).

— ZIP-fájl.ZIP (az előzőleg összepakolt archív állomány).

A PKSFX.PRG-nek az aktuális könyvtárban vagy az elérési útvonalon (Path-on) kell lennie. Ekkor az SFX-állományt elkészítő parancs:

zip2exe zip-fájl [sfx-fájl]

Ez az SFX a PKUNZIP majdnem minden parancsát elfogadja és végrehajtja. (A -v parancsokat nem.) Az állománynövekedés elég nagy, kb. 15,5 kilobájt. Az SFX archív felfrissítése azonos a ZIP-fájl kezelésével, csak a kiterjesztést is be kell írnunk.

Melyiket miért

LHA-SFX (LZH-fájl SFX-fájl)

Az LHA segítségével előállított SFX-állományok mind közül a legtömörebbek. A Telop egyedülálló lehetőséget ad, hogy kibontás előtt elegendő információt kapjon a felhasználó arról, kell-e neki az a csomag. Ha nem felejtettük el az SFX előállításakor az s parancs mellett a /x kapcsolót is megadni, akkor a !.bat indító állomány a /! parancs hatására közvetlenül a kipakolás után végrehajtásra kerül. Az SFX-nek megadhatunk egy célkönyvtárat is, ahova kicsomagolhatja magát. Nem utolsó szempont, hogy az SFX elkészítéséhez — a forrásállományokon kívül — mindössze a 34 K méretű LHA.EXE állomány kell.

PKZIP-SFX (ZIP-fájl SFX-fájl)

Hacsak nem a -j paranccsal készítettük az SFX-et, az önkibontó állomány a PKUNZIP minden parancsát ismeri (a -v parancsok kivételével). Így aztán a ZIP-SFX-ből akár nyomtathatunk is, sőt az 1.10 verziótól a jelszavas titkosítás és a listafájl használata is megengedett. Majdnem egyedülálló szolgáltatása (csak a ZOO-nak van hasonló), hogy a sérült archívokból az ép részeket ki lehet ollózni. A ZIPDMP segédprogrammal a kelleténél nagyobbra sikeredett ZIP-SFX-eket kisebbekre lehet szeletelni.

PAK-SFX (PAK-fájl, ARC-fájl, ZIP-fájl SFX-fájl)

A PAK programmal a SEA-ARC, a PKARC/PKPAK a régebbi verziójú PAK által előállított *.ARC állományokat könnyedén átkonvertálhatjuk egy PAK-SFX fájlba, csakúgy, mint a PKZIP-pel előállított ZIP-fájlokat. A PAK-SFX előállítása igen egyszerű, mivel a forrásállományokon és a PAK.EXE programon kívül semmiféle egyéb segédprogramra nincs szükségünk. A PAK-SFX tömör, gyorsan kinyílik.

ARC-SFX (ARC-fájl SFX-fájl)

Az SFX ugyanúgy kezelhető, mint az eredeti *.ARC-fájl az ARC programmal. Az előállított SFX által értelmezett parancsok megegyeznek Vernon Buerg ARCE programjának parancsaival.

PKPAK-SFX (ARC-fájl SFX-fájl)

A PKARC és a PKPAK programokkal előállított *.ARC-fájlon kívül a PKSFX.PGM állományra is szükségünk van a PKPAK-SFX előállításához. Ez utóbbit a PK361.EXE-ből a belőle kibontott MAKESFX.COM hozza létre. Az SFX-et a DOS-ból vagy a SHEZ hatékony közreműködésével állíthatjuk elő. Az egyik legnagyobb SFX fej rakódik rá az .ARC állományra, de cserébe a PKUNPAK/PKXARC minden parancsát értelmezni tudja az SFX.

LHARC

Kicsi a bors, de erős

Az LHARC tömörítő egy japán programozó, Haruyasu Yoshizaki (Yoshi) alkotása. Megírásakor olyan programot akart létrehozni, amely felveheti a versenyt az addigi legelterjedtebb s legjobbnak tartott tömörítőprogramokkal, a PKPAK-kal, a PKZIP-pel és a PAK-kal. Nyugodtan állíthatjuk, hogy ez sikerült is. Az LHARC tömörítési hatásfoka meghaladja a nagy vetélytársakét, s csak sebességben és néhány szolgáltatásban marad el a legjobbaktól.

A program 2.00 felett az LHA.EXE nevet viseli. Az előző verziók névmizériái után Yoshi remélhetőleg már megmarad ennél a névnél. A program parancsai és paraméterezései szinte teljesen kompatibilisak az előző verziókkal, így az itt leírtak nagy része az 1.13C változatra is igaz.

A program használata

Az LHA.EXE különlegessége, hogy egyetlen, 33479 bájt hosszúságú állomány biztosítja a bepakolást, a kicsomagolást és az önkinyitó archív készítését. További helyet takaríthatunk meg, ha az LHA.EXE programot is összenyomjuk az LZEXE vagy a PKLITE segítségével. Így egy kb. 24 kilobájtos programot kapunk, amely eddigi tapasztalataim szerint hibátlanul működik.

A programot leggyakrabban a DOS promptról használjuk, de az Errorlevel változó figyelésével batch-fájlokból is indíthatjuk. A ZIPVIEW is ismeri, a SHEZ az 5.9 verzióban még nem tud vele mit kezdeni.

A parancssor hat munkaterületre osztható:

1. LHA

A programnév. Legjobb, ha a Pathon, vagy az aktuális könyvtárban van. Szükség esetén a névben megadható az elérési útvonal is.

2. Parancs

Az LHA-nak tizenkét egybetűs parancsa van, amelyek közül egyszerre csak egyet írhatunk a parancssorba, méghozzá közvetlenül az LHA után, legalább egy szóközzel elválasztva attól. A parancsnak ez a rögzített helye a parancssorban. Ha nem írunk parancsot, vagy ha az LHA nem tudja értelmezni a parancs helyén álló karaktert, akkor bejelentkezik a program Help-képernyője, a programhasználat, a parancsok, a kapcsolók ismertetésével.

Az LHA által használt parancsok: a/f/u/m/l/v/t/e/p/x/d/s.

3. Kapcsolók (módosítók)

A program parancsainak hatását, működését befolyásoló paraméterek. A kapcsolók mindig egy bevezető / vagy – karakter után állnak. Nem kötelező kapcsolókat megadni, olyankor a parancsok alapértelmezése érvényes (lásd később). A kapcsolók állhatnak külön-külön is, egy-egy bevezető karakter után, de össze is vonhatjuk őket egy közös / vagy – jel mögé. Az LHA tizenöt értelmezett kapcsolójának további módosító paraméterei is lehetnek. A kapcsolók helye a parancsokkal ellentétben nem rögzített, azok a parancssorban — a parancs után — bárhol állhatnak, tetszés szerinti számban (amennyit a DOS elvisel). Az LHA.EXE alkalmazható kapcsolói: r, w, x, m, p, c, a, z, t, h, o, n, i, l, –.

A módosítók az LHARC kapcsolóinak működését befolyásoló paraméterek. Értékük általában 0, 1 és 2 lehet. A kapcsolók egy részénél az 1 és 2 teljesen azonos hatású. Gyakran a – és a + jelet is alkalmazhatjuk az adott kapcsoló be-, illetve kikapcsolására. A /z és a /w kapcsolóknál szöveges módosító is lehet.

4. Archív

A tömörített állomány. Kiterjesztést nem kell írnunk, az alapértelmezés .LZH (LZH-fájl), kivéve, ha önkibontó állományt (auto LZH-fájlt) akarunk az LHA-val készíteni, vagy ha szándékosan adunk meg más kiterjesztést. Ha nem adunk meg archív nevet, akkor a program hibaüzenetet közölve leáll és visszakerülünk a DOS-ba vagy a hívó batch-fájlba. Az önkinyitó archív ugyanúgy kezelhető, mint az LZH-fájl (ilyenkor persze a kiterjesztést is meg kell adnunk), de a bepakoláshoz és az aktualizáláshoz néha célszerűbb előbb kibontani, átszerkesztgetni, végül újra összecsomagolni.

Az archív neve elérési utat is tartalmazhat. Az archív megadásakor használhatjuk a DOS jokerkaraktereit (?, *), ha az e, x, t, pl, és a v parancs egyikét használjuk.

5. Bázis

Tömörítéskor a bázis a forrásállományokat tartalmazó könyvtár, kibontáskor a bázis a célkönyvtár. Nem azonos az átmeneti munkakönyvtárral. A bázis utolsó karaktere kötelezően a \ karakter. Egy parancssorban több báziskönyvtárat is megadhatunk, de kibontáskor egy adott fájl csak egy célkönyvtárba nyitható ki. A báziskönyvtár alapértelmezése mindig az aktuális könyvtár, ahonnan az LHA-t meghívtuk.

6. Lista

Itt fájl-maszkot adhatunk meg, és az LHA csak az annak megfelelő állományokkal foglalkozik. A maszk tartalmazhat DOS jokerkaraktereket (? és *), valamint Path-t is. Tetszés szerinti mennyiségben adhatunk meg maszkokat, ebben csak a DOS a korlát. A lista lehet Listafájl is egy bevezető @ karakterrel.

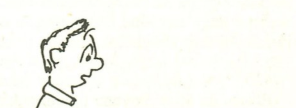
A program hibaüzeneteit, képernyőkimeneteit a DOS >, illetve > parancsaival átirányíthatjuk fájlba vagy nyomtatóra.

Az LHARC parancsai

Az LHA zömmel a tömörítőprogramok szabvány parancsait és parancselneve-







A verhetetlen csomagolóprogram

zéseit használja. (Zárójelben a parancs angol neve.)

Becsomagoló parancsok:

a — (Add) Az archívhoz hozzáad és bepakol. Új archívot is létrehoz. Kapcsolók nélkül használva rákérdezés nélkül felülír minden, az archívban levő azonos nevű állományt.

(LHA a c:\kakukk\tojas a: *.* c:\vacak\ *.doc)

u — (Update) A listában szereplő állományokkal aktualizálja az archívot. Új állományokat is becsomagol. Ha a már bent lévő esetleg frissebb, alapértelmezésben nem írja felül. Új archívot is tud készíteni.

(LHA u c:\doksi\pakolo a: *.doc)

f — (Freshen) Megegyezik az Update paranccsal, de nem tud új állományokat bevinni és új archívot létrehozni.

(LHA f c:\doksi\pakolo a: *.txt)

m — (Move) Megegyezik az Update paranccsal, de sikeres bepakolás után kitörli a forrásállományokat. Azokat az állományokat nem törli, amelyekkel nem frissítette az archív tartalmát.

(LHA m c:\doksi\pakolo e: *.*)

Kipakoló parancsok:

e — (Extract) A legegyszerűbb kipakoló parancs. Nem foglalkozik az elmentett Path-szal és a szintén elmentett könyvtárszerkezettel, csak a /x, illetve /r kapcsolókkal együtt alkalmazva. Csak frissebbeket bont ki alapértelmezésként, ha kint már van egy azonos nevű állomány.

(LHA e LHA210.EXE c:\doksi !)

x — (eXtract) Teljes visszaállítást végez, figyelembe veszi az elmentett Path-t és visszaépíti az eredeti könyvtárszerkezetet is. Egyebekben megegyezik az e paranccsal.

(LHA x a:dbase c:\db3)

p — (Print) Az LHA Print parancsa a kibontott állományokat a képernyőre bontja ki. Csak text állományokkal érdemes így használni. A nyomtató felé a DOS parancsával küldhetjük (LHA p RAJZOK.LZH prn). A nyomtatót nekünk kell a megfelelő üzemmódba kézzel átkapcsolni, mert a PKZIP-pel ellentétben ez a program erre a plusz szolgáltatásra még nincs felkészítve.

t — (Test) Csak szimulálja a kipakolást és közben teszteli a tömörített állományok CRC értékeit. Ha hibát talál, jelez. Az önkicsomagoló állományok (auto LZH-fájlok) tesztelésekor a kiterjesztést is meg kell adnunk. Ha magát az LHA.EXE programot teszteljük (LHA t LHA.EXE), akkor az alábbi üzenetre számíthatunk: "This file seems to be ORIGINAL distributed from H.Yoshi."

Törlő parancs:

d — (Delete) A parancs a megadott állományok nélkül építi újjá az archívot.

(LHA d doksi *.bak *.tmp)

Tartalomjegyzék-parancsok:

l — (List) Az archív tartalomjegyzékét listázza ki, az önkinyitóéval együtt. Az alábbi adatokat láthatjuk benne: Name = a becsomagolt fájl. Original = tömörítés előtti méret. Packed = tömörített méret. Ratio = az eredetinek hány %-a lett. Date, Time = keletkezési időpont. Attr = az adott fájl attribútuma (h–hidden, s–system, w–writable, r–readonly, a–archive). Type = becsomagolás típusa. CRC = ellenőrző összeg. Ha a tömörített fájl Path-szal lett elmentve, akkor a név előtt egy + jel áll. Alapértelmezésben egy fájl adatai egy sorban vannak. A /x kapcsolóval külön sorban a fájlok Path adatai is megjelennek.

v — (View) Megegyezik az l parancs /x opcióval együtt való használatával.

Önkibontó fájl készítése:

s — (Self-extract) Az előzőleg összeállított LZH-fájlból egy önkicsomagoló auto LZH-fájlt csinál. A parancs alapértelmezésként a butább (és kisebb, kb. 1,5 K) Small model önkinyitót készíti el. Ha /x kapcsolót is használunk, akkor a nagyobb (és okosabb, kb. 1,9 K) Large model-t készíti el az LZH-fájlból az LHA. (Vö. a 10. oldallal.)

Az LHARC kapcsolói

/x[0|1]

A teljes nevek (eXtended) használatát is engedélyezi. Az /x0 és az /x-kikapcsolja. Az s parancs mellett a Largemodel önkinyitó állomány építését biztosítja.

/p[0|1]

Kibontáskor a megfelelő Path-ra teszi a fájlokat (precízen).

/c[0|1]

Az u, f, m, e és x parancsok alkalmazásakor automatikusan felülír, és nem nézi a keltezéseket.

/m[0|1|2]

Az e parancs mellett használva, ha van a kibontandóval azonos nevű fájl a célkönyvtárban, akkor az /m2 kapcsoló beállításával nem üzenetet (Message)

küld, hanem egy 000 és 999 közötti, még nem használt kiterjesztéssel csomagolja ki az állományt.

/a[0|1]

Az LHA alapértelmezésként NEM archiválja a rejtett (hidden), a rendszer- (system) és a csak olvasható (read only) állományokat. A /a1 kapcsoló engedélyezi ezen állományok archiválását is. Kibontáskor az LHA csak akkor állítja vissza az elraktározott attribútumot, ha /a1 kapcsolóval használjuk.

/r[0|1|2]

/r0 — Csak a megadott állományokat csomagolja (nem rekurzív).

/r1 — Az alkönyvtárakból is pakol, de a szerkezetet nem tárolja el.

/r2 — Az alkönyvtárakból is pakol, és a szerkezetet is eltárolja.

/t[0|1]

A bepakoló és a törlő parancsok mellett használatos, az új archívok keltezését (Time) állítja:

/t0 — a rendszeróra állásához (alapértelmezés),

/t1 — az archívban lévő legfrissebb állományhoz.

/w[0|1|+|könyvtár]

Az átmeneti munkakönyvtár (Work directory) megadására szolgál ez a kapcsoló. Az alapértelmezés általában az a könyvtár, ahová az új archív kerül.

Ha /w+, vagy /w1 kapcsolót használunk, a munkakönyvtár az a könyvtár lesz, ahonnan az LHA-t meghívtuk.

Ha a /w után könyvtárnevet is adunk, akkor azt szóköz nélkül, \ jellel lezárva kell megadni (/wc:\átmeneti\).

Ha lemezeken levő archívokat aktualizálunk, célszerű a winchesteren, vagy Ramdrive-on tartani az átmeneti állományokat.

/n[0|1|2]

Az LHA működését nyomon követő kiírásokat letilthatjuk vagy engedélyezhetjük:

/n0 — nem tiltja le a kijelzéseket,

/n1 — a folyamatkövető ooo... jelek kiírását letiltja,

/n2 — a fájlnevek és a tömörítettség kiírását tiltja le.

/z[0|1|2|ext.]

Tömörítés nélküli (Zero compression) archiválást tesz lehetővé:

/z0 — Mindent tömörít bepakoláskor (alapértelmezés).

/z1 — Csak bepakol, tömörítés nélkül.

/z2 — Archívokat (ARC / PAK / LZH / LZS / ZIP / ZOO) nem tömörít.

/z — A kiterjesztés nélküli állományokat nem tömöríti.

/z[ext] — Az adott kiterjesztésű állományokat nem tömöríti. Ez utóbbi kapcsoló többször is megadható egy parancssoron belül.

/o[0|1]

A régi (Old) LHARC 1.13-mal kompatibilis fájlt csinál. (Az 1.13 verzió azonban sajnos nem tudja kibontani). Használatakor a header level automatikusan beáll h1-re.

/h[0|1|2]

A fejrovat (Header) alapértelmezése /h1.

/i[0|1]

A DOS alatt csak akkor érdekes, ha nem DOS rendszerben készített archívokat akarunk kibontani.

/l[0|1|2]

A fájlnevek megjelenítési formáját befolyásolja be- és kicsomagoláskor:

/l0 — Csak a fájlnevek.

/l1 — A teljes tárolt, illetve tárolandó nevek, két sorban.

/l2 — Az éppen feldolgozott állományoknak is a teljes neve jelenik meg.

/–[0|1]

/–1 — A – és a @ karaktereket a fájlnév részének tekinti.

/–0 — A – kapcsoló bevezetőkarakter, a @ pedig listafájl-jelölő karakter (alapértelmezés).

Az LHA környezetleíró változói

LHA

Ebben a változóban tárolhatjuk az LHA kapcsolóinak alapértelmezéseit. Pl. az AUTOEXEC.BAT-ban: SET LHA=/Wc:\temp\ /X1R2A1L2Z0

TMP

Ez a változó is beállíthatja az LHA munkakönyvtár alapértelmezését. Ha használjuk, akkor a /w kapcsoló szinte felesleges. A változót más programok is használják (pl. a Microsoft Word).

Listafájl

Használata célszerű, mert dokumentálhatjuk, hogy mikor, mivel és hogyan módosítottuk az archívot. Ha egy fájlnevet az archív után @ karakterrel kezdünk a parancssorban, akkor azt az LHA listafájlnak veszi. A listafájl közönséges ASCII szövegfájl, amelyben egyszerűen felsoroljuk mindazokat a kapcsolókat, módosítókat, báziskönyvtárakat, fájlmaszkokat, amelyeket a DOS parancssorába írnánk, ha annak mérete nem lenne korlátozott. A listafájlban szóközt használunk elválasztó karakternek. A CR kocsivissza (<ALT-13>) jeleket az LHA figyelmen kívül hagyja. A listafájl tetszőleges szövegszerkesztővel szerkeszthető, ha az normál ASCII-állományokat készít. (Mazochistáknak ajánlom a DOS EDLIN programját.)

Az auto LZH-fájlok

Small model

A tömörített LZH-fájlnál 1,5 K-val nagyobb, csak a –o parancsot ismeri, Path információkat nem tartalmaz, s végül: nem tudja a !.BAT-nak átadni a vezérlést.

Large model

Mintegy 1,9 K-val növeli meg a forrás LZH-fájl méretét. A /! kapcsolóval a belőle kibomló !.BAT fájlra átkerül a vezérlés. Megadható a célkönyvtár is. A szerkezet visszaépítése itt alapértelmezés, de a /x kapcsolóval letiltható.

Telop (!)

A Telop (!) egy üzenetfájl, amit az archív legelejére kell becsomagolni. Tetszőleges ASCII szövegszerkesztővel szerkeszthető állomány.

Nem érdemes egy képernyőnél hosszabb szöveget beletenni, mert úgyis gyorsan kigördül az eleje a képernyőről. Célszerű a bepakolt állományokról, a kibontás módjáról, a szerzőről stb. pár sort elhelyezni benne.

Bepakoláskor mindig elsőnek kell betenni az archívba, különben nem ő jelentkezik be elsőként, hanem vár a sorára, amíg az előtte becsomagolt állományok kibomlanak.

!.BAT

Ebben az állományban helyezünk el minden olyan paramétert, amely a tömörített program kipakolás utáni automatikus elindításához szükséges.

A visszaadott Errorlevel-értékek

0 — Normál futás hiba nélkül.

1 — CRC hiba, főleg az e, x, t parancsok esetén szokott előfordulni.

2 — Végzetes hiba történt, a folyamat leállt.

3 — Nem tudta a program az átmeneti LHTNP)2(.LZH állományt átmásolni vagy átnevezni. A munkakönyvtárban ott az átmeneti fájl.

PKLITE
Futtatható állományok tömörítése

Phil Katz cége a nagy sikerű PKARC, PKPAK és PKZIP után egy érdekes új tömörítővel, a PKLITE rendszerrel jelentkezett. A futtatható állományokat tömörítő program az állományok kódját nagymértékben módosítja, a forráskód visszafejtése pedig nehéz, mert több algoritmus alapján választja ki a kódoló eljárást és azokra nem utal az állományban.

A programot a PKLTE10.EXE önkicsomagoló program tartalmazza. Annak elindítása után — a PKWARE termékeitől megszokott módon — bomlik ki a program, az angol nyelvű dokumentációs állományokkal együtt. A PKLITE változatlan formában szabadszoftverként továbbadható, de professzionális célra nem használható fel, arra csak a kereskedelmi forgalomban beszerezhető, irreverzibilis kódolást is alkalmazó regisztrált változat szolgál.

Használatának feltételei

Tömörítési hatásfoka azonos a PKZIP-pel. A tömörített állomány futtatható marad, a memóriában pakolja ki magát. A tömörített állomány futása során az összes dekódolási művelet a háttérben, a felhasználó tudta nélkül, automatikusan történik. Tudomásul kell venni azonban, hogy a PKLITE nem alkalmazható mindegyik program összenyomására. Sok program azonban ily módon nem működik. Hogy melyik, azt csak a tapasztalat dönti el, illetve ha magunk írunk programot, eleve figyelembe vehetjük működtetésének feltételeit:

— Nem alkalmazható MS-Windows alá írt programoknál.

— Nem alkalmazható semmilyen másolásvédett programnál.

— Nem alkalmazható olyan programnál, amely magában hordja overlay ágait is. (Például FoxPro, Microsoft Codeview, Microsoft C2 „C" compiler, FSD, Turbo Debug stb.) Ilyenkor a program „overlay not found" hibaüzenettel leáll. Ha viszont az eredeti változatot látja, akkor továbbmegy...

— Nem alkalmazható az OS/2 környezetben futó programoknál.

— Nem alkalmazható akkor, ha előtte az állomány becsomagolására már egy másik futtatható állománytömörítőt használtunk. (Például EXEPACK, LHZEXE stb.)

A PKLITE program a DOS parancssorból kiadott utasításokkal kommandírozható. Rossz paraméterezés esetén megjeleníti a Help képernyőjét, ahol az opciók használatát megnézhetjük. Egy parancssor, ha minden opciót megadunk neki, a programozásban megszokott általános leírást használva a következőképpen néz ki:

PKLITE [options] [d:][path] Infile [[d:][path] Outfile]

ahol
PKLITE = Az elindítandó PKWARE program neve.

Options = Opciók.
[d:] [path] Infile = A be- vagy kipakolandó futtatható állomány neve és elérésének útvonala.
[d:] [path] Outfile = A célállomány neve, és annak a helynek teljes elérési útja, ahova tenni akarjuk. Ha az aktuális könyvtárban helyezzük el, akkor mindkét esetben elég az állománynevek feltüntetése.

Példa legegyszerűbb alkalmazására

C:\DOS> PKLITE attrib.exe
Ekkor az attrib.exe állományt az aktuális könyvtárban keresi, onnan felszedve összenyomja és az új verzióval felülírja a régit. A képernyőn megjelenő üzenet:

PKLITE ™ Executable File Compressor.
Version 1.00 Copyright 1990
PKWARE Inc. All Rights Reserved.
Compressing attrib.exe
Original Size: 10656
Compressed Size: 6790
Ratio: 36.3

Ha célállományt is megadunk a programnak, akkor az állomány-információ ezt is tartalmazza:

Compressing LIGHT.EXE into file A:SMLIGHT.EXE

Gyakorlati tanácsok

— Teljesen mindegy, hogy a parancsok és az állománynevek begépelésénél kis- vagy nagybetűket használunk, és esetleg keverjük is őket.

— A parancsok opcióit (vagy más néven a kapcsolókat) a „–" (<Alt-45>) karakterrel, vagy pedig a DOS hagyományosan opciójelzőként alkalmazott „/" karakterével használhatjuk.

— Az egyes opciók kombináltan is megadhatók: például „–o –b" vagy pedig „–ob".

— Ha nem adunk meg kiterjesztést, akkor a PKLITE automatikusan minden olyan állományt tömörít az aktuális könyvtárban, amely az általunk megadott nevű, továbbá .COM és/vagy .EXE kiterjesztésű. Az állománytípust automatikusan felismeri. Például ha az adott könyvtárban azonos néven egy .COM és egy .EXE állomány van, akkor mindkettőt bepakolja.

— Ha nem adunk meg célállományt, akkor a tömörített változattal automatikusan felülírja a régit!

Figyelem! Nagyon fontos — különösen a regisztrált program esetében —, hogy az eredeti programlemezt soha ne használjuk. Készítsünk másolatot, s arról installáljunk, arról dolgozzunk. Figyeljünk arra, hogy vírusmentes környezetben dolgozzunk, mert a program a vírussal fertőzött programot is tömöríti, s az így kapott programban a vírus — ha az irreverzibilis professzionális kódolást alkalmaztuk — semmilyen eszközzel nem mutatható ki, csak a program futtatásakor derül ki, hogy szüli a vírust!

Kis János

A kapcsolók hasznáról

Könnyebb velük ZIP-elni

Amikor pár évvel ezelőtt először futtattam le a PKWARE PKZIP programját és a kapcsolók felsorolása nem fért ki egy képernyőn, bizony megijedtem.
Ma pedig már annyira megszoktam őket, hogy az LHA helyett — amely pedig kicsivel jobban tömörít —, továbbra is inkább a PKZIP-et használom, mert annak hasznosabb kapcsolói vannak.

A PKZIP programhoz egy 140 kilobájtos angol nyelvű kezelési utasítás is tartozik, de azok kedvéért, akik ezt nem tudják (vagy nem akarják) elolvasni, bemutatjuk a legfontosabb kapcsolókat. A PKZIP használata igen egyszerű:

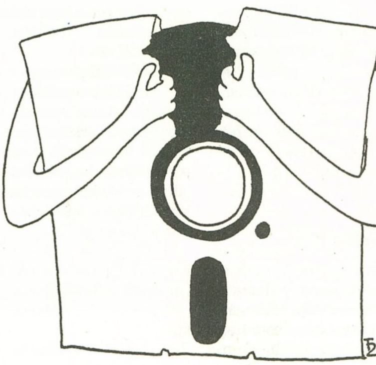
PKZIP [–b[path]] [options] zipfile [@list] [files...]

Érdekes a külön kiemelt –b kapcsoló, amellyel megadható, hogy a PKZIP hová helyezze el munkafájlját. Ez igen jól jön, ha egy floppy-n lévő .ZIP fájlhoz akarunk hozzáadni újabb fájlokat. A leghasznosabb kapcsoló az újabb fájlok hozzáadását végző –a, bár ezt legtöbbször nem kell kiírni. Ennek a párja a –d, amellyel viszont fájlokat lehet törölni. A fájl tartalmát a –v-vel nézhetjük meg. Hasonló kapcsolója a PKUNZIP-nek is van.

A nagyobb szoftverek jellegzetes könyvtárszerkezetben helyezkednek el. A könyvtárak nevét is becsomagolhatjuk a .ZIP fájlba a –P kapcsolóval. Ha az –r kapcsolót is megadjuk, akkor például a PKZIP –r –P tc c:\tc*.* becsomagolja a teljes Turbo-C-t, összes könyvtárával és programjával együtt. A kicsomagolás PKUNZIP –d tc parancssorral történik. Hasznos tulajdonsága a PKZIP-nek, hogy fel lehet sorolni azokat a fájlokat is, amelyekre nincs szükségünk. Ez különösen jól használható archiváláskor. Ha a PKZIP –bc:\ –P –x*.BAK A:\WORK C:\WORK*.* parancssort minden nap végrehajtjuk, akkor az összes munkafájlunkat tömörítve elmenthetjük, és a .BAK fájlok sem foglalják a helyet. Ha így elvisszük a programot egy másik gépre, ott PKUNZIP –d A:\WORK beírása után tovább dolgozhatunk. Hasonlóan egy C-ben programozónak nincs szüksége a fordító által létrehozott object- és listafájlokra, programjait a PKZIP –xC:\ –x*.BAK –x*.OBJ –x*.LST –x*.MAP A:\PROGRAM C:\TC\PROGRAM*.* utasítással mentheti le. Ha ez az egy sor egy batch-fájlban van, akkor használata is igazán kényelmes. Ha olyan sok fájlt akarunk felsorolni, hogy azt a DOS már nem szereti, akkor írjuk azokat egy fájlba és annak a fájlnak a nevét adjuk meg a @ után. Ez a –x kapcsolóval együtt is működik.

És mint tegyünk, ha a kapott .ZIP fájl nem fér fel egy floppyra?

A floppymellékleten van egy SLICE nevű program, amely a hosszú állományokat szétvágja darabokra. Az ugyan nem derül ki, hogy ki írta, de rendkívül hasznos. Használatánál azonban veszély is leselkedhet ránk: ha a fájl neve hat karakternél hosszabb, a program könnyen kiakad.



LHArakiri japán önkicsomagoló

A tömörítőkről közölt összeállításunkkal kapcsolatos észrevételeiket kérjük közvetlenül a szerzőhöz eljuttatni:
Nagy Gábor
1116 Budapest XI., Kisköre u. 16.
Tel.: 166-1444 (18–24 óra között)
157-5311/160 (9–16 óra között)

Közvéleménykutatás

A Mikroszámítógép Magazin nyomdokain — de teljesen új koncepcióval — 1990. júniusában jelent meg először az Alaplap. Egy év alatt kialakult a lap jelenlegi szerkezete, jellege, stílusa, arculata. Szerkesztési munkánk javításához szeretnénk most a lapról Önökben kialakult véleményt minél részletesebben megismerni, ezért mellékeltünk egy közvéleménykutató kérdőívet. Kérjük, töltsék azt ki, és küldjék vissza az Alaplap szerkesztőségébe. (1251 Budapest, Postafiók 71.) Akik válasza 1991. augusztus 31-ig beérkezik, azok jutalomsorsoláson vesznek részt. Fődíj: 1 db IMB-kompatibilis AT számítógép!

ZIPVIEW

„Egy híját esmértem..."

A Norton Commandert (NC-t), mindenki ismeri, aki már kicsit is beleszagolt a PC-k világába. A programcsomag igen hasznos részei a bepillantó állományok, amelyekkel táblázatokba (123VIEW), adatbázis-állományokba (DBVIEW, RBVIEW, PARAVIEW, REFVIEW), szövegekbe (WPVIEW) és grafikai állományokba (PCXVIEW) lehet betekinteni, az adott programba való belépés nélkül. Lehet, hogy Csokonai nem erre gondolt, amikor leírta: „Egy híját esmértem örömimnek még"..., de nekünk a tömörített állományokba való bekukucskálás lehetősége hiányzott a Norton Commander arzenáljából. Most itt van!

Sok vetélytársát megelőzve egy ügyes holland programozó, F.A. Oldenhuis (Robert van Hoeven munkáit felhasználva) megírta a ZIPVIEW programot, ami szervesen beleilleszkedik az NC-csomagba. PC-tulajdonos és -felhasználó barátaim a ZIPVIEW programot megismerve azonnal betelepítették azt gépük \NC könyvtárába a Norton Commander mellé.

Mit tud az apróság?

Elég sok kellemes szolgáltatása van:

— Felismeri és kilistázza a legelterjedtebb tömörítő programok által előállított archív állományok (.ZIP, .ARC, .LZH, .PAK, .ZOO, .DWC) belső tartalomjegyzékét (név, tömörített és eredeti méret, tömörítettség, CRC). Sajnos a többszörösen pakolt állományok néha megzavarhatják.

— Ugyancsak felismeri a fenti archívokból készített önkicsomagoló archívokat, és azoknak is kiírja belső tartalomjegyzékét, ha meghíváskor az SFX-fájl kiterjesztését is megadjuk.

— Ha az adott archívfájl kicsomagolója path-on van, akkor az <F3> lenyomásával a kurzorral kiválasztott fájlt kinyitja vele és kiírja a képernyőre. Megtekintés után az átmenetileg létrehozott fájlt törli.

— A kijelzett archív tartalomjegyzék állományai közül kedvünkre kijelölhetünk az Insert gomb, valamint a szürke (a numerikus billentyűzeten lévő) + és – segítségével. A kiválasztott állományokat pedig a ZIPVIEW ki is nyittatja a path-on levő, megfelelő kibontók segítségével. Nekünk csak azt kell még beírnunk, hogy hová pakoljon ki. (Ha nem adunk meg célkönyvtárat, akkor az aktuális könyvtárat választja.) A kibontó program megfelelő meghívása már nem a mi dolgunk, ezt elvégzi helyettünk a ZIPVIEW. Nem semmi, ugye?

Hogyan használjuk?

A ZIPVIEW kipakoló segédprogramként való használatához négy előfeltétel szükséges:

— A megfelelő kibontók elérési útvonalra (path-ra) állítása.

— A ZIPVIEW meghívása után a kibontandó állományok kijelölése.

— A kicsomagolás helyének megadása.

— A célkönyvtárban elegendő hely biztosítása.

DOS promptról az ZIPVIEW-t például így indíthatjuk:
C:\ZIPPEDZIPVIEW kakukk.ZIP

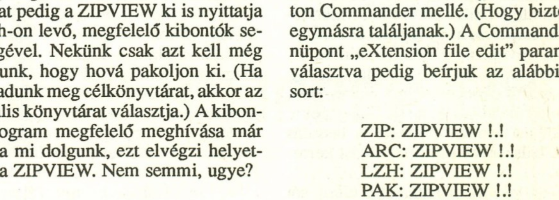
A jelenlegi, 1.0 verzió az archív nevében nem fogad el ú.n. joker (? és *) karaktereket. Mind a nevet, mind a kiterjesztést be kell írni. A Norton Commanderben ezért is könnyebb dolgozni. Az egyik ablakban legyen az a könyvtár, ahol a ZIPVIEW.EXE van. Nyillal odaállunk, majd <Ctrl-J> leütésével a fájlnevet a képernyő alján levő parancssorba küldjük. Egy szóköz után menjünk át a másik ablakba és a megtekintendő tömörített állománnyal ismételjük meg a fenti műveletet. Tehát egyetlen nevet sem kell begépelnünk a parancs kiadása előtt.

Még kényelmesebben használhatjuk egy kis előmunkálattal. Betesszük a ZIPVIEW-t az \NC könyvtárba a Norton Commander mellé. (Hogy biztosan egymásra találjanak.) A Command menüpont „eXtension file edit" parancsát választva pedig beírjuk az alábbi pár sort:

```
ZIP: ZIPVIEW !.!
ARC: ZIPVIEW !.!
LZH: ZIPVIEW !.!
PAK: ZIPVIEW !.!
ZOO: ZIPVIEW !.!
DWC: ZIPVIEW !.!
```

majd elmentjük a változtatást a Nortonban megszokott <F2>-vel. Ezek után már bármikor csak rá kell állnunk a megnézendő archív nevére és az Enter-t lenyomni.

Természetesen az .EXE és .COM kiterjesztésű önkinyitó archívokat így nem tudjuk megnézni, azokhoz be kell pötyögnünk a DOS-promptról a szükséges adatokat (lásd fent).



SFX-állomány

Stacker — pro és kontra

Fújd fel a winchestered!

Teli van a... nemcsak a hócipőnk, hanem a winchesterünk is. Annyira, hogy nyertes lottószelvényünk beváltása előtt is kellene találni valamilyen átmeneti megoldást. Például „felfújni" a merevlemezt. Ehhez az Alaplap múlt havi (1991/6.) számában már ismertettük a NewSpace nevű shareware-megoldást. Most bemutatunk egy drágább változatot, a Stackert.

A Stacker koprocesszor-kártyás változata elég drága (39 000 Ft), inkább vállalatoknak ajánlható, de a szoftveres verzió könnyebben elérhető (jelenleg 19 900 Ft).

Pro

1. Átlagban megduplázza a winchester kapacitását, anélkül, hogy növelné a lemezreírás fizikai sűrűségét. Működésének alapelve, hogy a Stacker használatába bevont meghajtókra mindig tömörítve ír, és onnan kicsomagolva olvas, az AT-gépeken szinte észrevehető lassulás nélkül. (A lassú XT-gépekre sajnos ez nem áll, ezeknél a lassulás miatt célszerűbb más megoldást keresni.)

2. A tömörítés hatásfoka eléri, sőt néha meghaladja a PKZIP-ét, viszont az installálás után nem kell paramétereznünk.

3. Nem kell időt és energiát rabló ki- és becsomagolással vesződni, mivel a Stackeren keresztül a tömörített állományokat mindig kibontva látjuk.

4. A floppy-ra nem tömörít, tehát régi lemezeinket bátran használhatjuk és a Stacker nélküli gépekkel zavartalanul tudunk adatot vagy programot cserélni.

5. Pofonegyszerű az installálása, igen jó a kézikönyv, rengeteg hasznos tanáccsal a különböző gépekhez. Az installálás után többet nem kell paraméterezgetnünk, mindent automatikusan a háttérből intéz.

6. A lemez nem másolásvédett, ezért harverhiba, DOS-csere, vírusfertőzés vagy más ok miatt szükségessé váló rendszer-újraépítés akárhányszor elvégezhető.

7. A nagyméretű winchestereket a 3.30-as DOS alatt 16 megabájtonként tudja felfújni, így több partícival kell számolni. Megfelelő DOS (Tandon, Compaq, DR DOS, MS-DOS 4.00 felett) használata esetén tetszőleges méretű partíciókat (250 megáig) alakíthatunk ki az egybefüggő winchesteren. A 3.30-as DOS használatakor a Stacker a 16 MB-nál nagyobb partíciókat kétfelé osztja, a 16 MB-os részt és a maradékot két menetben duplázza meg.

8. Ha az installáláskor a SWAP opciót kértük, akkor nem kell átírnunk batch állományainkat, a Stacker udvariasan megcseréli a meghajtó-neveket.

9. Rezidensen csak 30 K a helyigénye, de partíciónként további 3 K-ra van szüksége. Ez megfelelő memóriakezelő programmal áttehető a 640 K feletti részbe, vagy az EMS-területre.

9. Installáláskor megválasztható, hogy csupán a szabad lemezterületet duplázza-e meg, vagy az inkrementális (bővítő) installációt választva a már winchesteren lévő programokat is fokozatosan pakolja be a Stacker által kezelt részbe. Ez utóbbi megoldás lassabb ugyan, de nem kell kimenteni, majd újra visszacsomagolni mindent, és az eredeti könyvtárszerkezet is megmarad.

10. Gyorsulhat az adatforgalom a merevlemezzel, mert a fizikailag kiírásra/beolvasásra kerülő rész a tömörítésnek köszönhetően kisebb, azonkívül az inkrementális installáláskor a program a sűrített állományokat egybefüggő, könnyebben elérhető fájlba zsúfolja össze.

11. Installáláskor kérhetjük, hogy egy cache terület kijelölésével tovább gyorsuljon a program. Mivel azonban ez a választás a szokványos 640 K-ból csíp le, célszerűbb külső (PC-Cache, NCache, stb.) programot használni, ami a 640 K felett, vagy az EMS területen foglal le helyet. A Stacker ezekkel is kiválóan együttműködik.

12. Megfelelő memóriamenedzsert használva a gyakran használt többi rezidens programmal együtt a HIMEM, vagy EMS területre helyezhetjük a Stacker-t is.

Kontra

A tisztesség kedvéért essen szó a veszélyekről is:

1. Ha valaki a kézikönyvben megadott minimálisan 1 megabájtnál kisebb szabad lemezterülettel kezd neki az installálásnak, elveszítheti a merevlemez teljes tartalmát.

2. Keveredést okozhat, ha installáláskor elfelejtünk SWAP-ot kérni. Ez manuálisan is beállítható, de minek.

3. A Stacker által nem értelmezett vagy nem kezelhető (pl. A:, B:) meghajtókat a Norton Commander 0 bájt szabad területtel és 0 bájt kapacitással jelzi ki. Ez persze csak a kijelzés, ettől még ugyanúgy írja és olvassa ezeket a lemezeket, mint a többit. A PathMinder és a PCTools jobban látja ezeket az adatokat.

4. A merevlemez meghatározott területére közvetlenül író egyes programok kárt tehetnek a Stacker által ugyanott összesűrített állományokban.

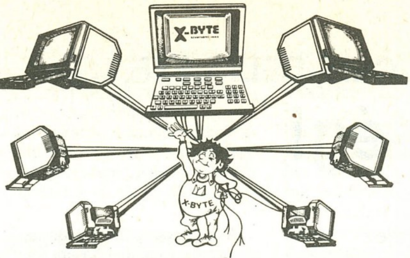
5. SWAP esetén a C:\-ben lévő AUTOEXEC.BAT-hoz nem szabad hozzányúlni. Az igazi AUTOEXEC.BAT a SWAP-olt meghajtón van, az igazi CONFIG.SYS-szel együtt.

6. Egyes nagy helyigényű programok nem férnek össze vele. A megoldás: DR DOS 5.0. Ekkor a driver a programokat és magát a Stacker-t is a 640 K terület fölé löki, és a DOS 32 megás korlátját sem kell kerülgetnünk.

7. DR DOS 5.0 mellett nem használható a SWAP funkció.

8. Azok a programok, amelyek szintén sokat vermelnek (pl. a Chessmaster 2000 sakkprogram) könnyen kiakaszthatják a rendszert, a Stacker-ral egyetemben.

Mindent összevetve a Stacker hasznos és jól kihasználható program, de tisztában kell lennünk korlátaival és meg kell tanulni hozzá a megfelelő bánásmódot.



a jövő most kezdődik!

X-BYTE®

SZÁMÍTÁSTECHNIKA

Törv. védve

SZÁMÍTÓGÉP HÁLÓZATOK

Kiváló minőség, közepes ár!

kiváló áruk fóruma

1138 Budapest, Népfürdő u. 17/E

Tel. és fax: 173-1232

Telex: 22-3399

Topreklám

INFORMÁCIÓKÉRÉS: 16 ▼

Canon Canon

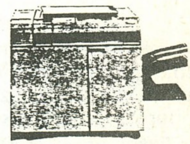
VÁLTSON SZÍNESRE

Canon

MÁSOLÓGÉPPEL!

CSÚCS AMIT TUD:

- Kicsinyítés
- Nagyítás
- Montírozás
- Tükörkép-készítés
- Képismétlés
- Poszter készítés
- 17 millió színáranyalat



Győződjön meg róla!

fénymásolás

TONER KFT

1095 Budapest, Mester utca 21.

Tel.: 113–1687, 134–3516

Canon Canon

INFORMÁCIÓKÉRÉS: 07 ▼

CANON FÉNYMÁSOLÓ

FELHASZNÁLÓK, VÁSÁRLÓK!

Magyarországon a legolcsóbban!!

Festékkazetták újratöltése eredeti USA technológiával,

garanciával (kék és barna színben is)

Budapesten ingyen házhozszállítás és géptisztítás

ÚJ FESTÉKKAZETTÁK

star Lézernyomtatók,

Canon fénymásolók

árusítása. Ingyenes szaktanácsadás

CompuDrug Standard Kft.

régi szolgáltató ÚJ HELYEN

1101 Budapest, Népliget, Planetárium

Telefon: 133-1576, 134-1164

Árusítás, újratöltés utánvétellel is!

LASER PRINTER

INFORMÁCIÓKÉRÉS: 23 ▼

Az új, bővített

CLIPPER 5.01

kapható

más, érdekes

Clipper kiegészítőkkel

(dGE, Netlib, dbdBAK, Tools II. stb.)

Kérjen részletes prospektust.

R-SOFT-SZENZOR

Budapest

Telefon: 201-6891

Fax: 201-8619

1277 Budapest 23, Pf. 45.

INFORMÁCIÓKÉRÉS: 17 ▼

COMP.COM, DIFF.EXE, COMPARE.COM, DELTA.EXE, ...

Mecsoda különbség!

Bár nem igazán sportszerű különböző súlycsoporthoz tartozó programokat összehasonlítgatni, most mégis ezt tesszük, hiszen az alapvető célkitűzés mindegyik program készítője számára ugyanaz volt. Az egyes megoldások felfogás és igényesség dolgában persze jelentősen eltérnek egymástól, nem utolsósorban attól függően, hogy ki mire taksálta az alapproblémát, és hogyan ítélte meg: mennyi várakozási időt bír el a gép mellett ülő felhasználó türelme.

Gyors eredmények nagyon egyszerű eszközökkel, nagyon rövid idő alatt elérhetők. Komolyabb eredmények több ráfordítást igényelnek, de az már egyedi megítélés kérdése, hogy meddig érdemes elmenni az időrabló számításokban a teljesítmény javítása érdekében.

A feladat — fájlok egyedi vagy csoportos összehasonlítása — fontos és viszonylag könnyű témának ígérkezett. Jó néhányan belevágtak a megoldásába, eleinte nem is gondolva, mekkora különbség van az olcsó, a drágább és a még drágább megoldások között. Van, aki talán nem is törekedett többre, mint hogy néhány kilobájtba beleszorítsa az alapprobléma korrekt megoldását: fájlok azonosságának vagy eltérő voltának megállapítását.

A „hogyan tovább" kérdése voltaképpen az eltérő fájlok esetében merül fel: kézenfekvő feladat az eltérések helyének, jellegének megállapítása, aztán jön sorra a különbségek szemléletes bemutatása, kigyűjtése, kinyomtatása és számos egyéb járulékos feladat. Mindez már egyre nehezebben fér bele egy viszonylag igénytelen segédprogram kereteibe. A feladatok körének egyre tágabb értelmezésével végül már szinte egész programrendszer kerekedett ki az egyszerű feladatból.

Mindent bele?

A „súlycsoportok" tehát ennek megfelelően alakultak, programhosszban mérve a 4-5 kilobájttól egészen a 240-250 kilobájtig. És ez a harmadfélszáz kilobájtos helyfoglalás már csak a többszöri méretcsökkentési akciók utáni maradék! Hiába, egy bizonyos határon túl öntörvényűvé válik a növekedés: ha már ez benne van, tegyük bele azt is, amazt is, ha már főprogram lett belőle, feltétlenül kezelje automatikusan az összes képernyőtípust, a kiterjesztett memóriáról nem is szólva... De ha már egyszer sok helyet foglal el, akkor biztosítani kell a DOS-on keresztül a kimenetet más programokhoz, persze mindazzal, amivel ez együtt jár: zsugorodjon össze a több száz kilóról ibolyaszerénységű kis rezidens programmá... és így tovább a végtelenségig.

Félretett gondolatforgácsok

A probléma fontossága és szinte mindennapos előfordulása nyilvánvaló. Kinek nem okoz újra és újra visszatérő gondot, hogy különböző állományainak többféle változata vagy több másolata keletkezik munka közben. Vagy azért, mert a rendszer gondosan megőrizte az előző állapotot, vagy mert óvatlanul, szándékaink ellenére más helyre íródott be valami, mint ahogy gondoltuk, vagy elfeledkezünk róla, hogy valahol már megvan, és újra bevisszük, vagy ezernyi más okból. Még azt sem mondhatjuk, hogy sajnos, mert időnként bizony igencsak jó szolgálatot tesz például egy .BAK kiterjesztésű fájlból előbányászható információ, amikor, mondjuk, helyre kell állítani egy véletlenül letörölt állományt. De félbehagyott programrészleteket, félretett gondolatforgácsokat is gyakran éppen a félkésztermékek között keresgélve idézhetünk fel.

A legegyszerűbb feladatra, a teljes azonosság megállapítására megbízható megoldást nyújtanak a „hagyományos" fájlösszehasonlító programok. Ezek közé tartozik a DOS COMP.COM programja, a PCTools COMP funkciója, és még néhány ismert, hasonló rendeltetésű, önálló egységként hívható vagy valamilyen komplett kiszolgáló rendszerbe beépített segédprogram. E hagyományos, talán „ellenőrző-hasonlító" programoknak nevezhető termékek fő funkciója, hogy karakterről karakterre végigbogarásszák, nem lelnek-e eltérő bájtokat valahol, valahányadik pozícióban. A mellékfunkciójuk pedig, ha ilyen is van, a tapasztalt eltérések pontos kilistázása.

Az efféle butácska programok is roppant hasznosak tudnak lenni, főleg olyankor,

— ha kétségeink vannak, hogy egy másolás valóban hibátlanul zajlott-e le;

— ha biztosak akarunk lenni, nem töröltek-e bele valamelyik, szerencsére másolatban is őrzött programunkba vagy adatállományunkba;

— ha tudni akarjuk, nem változott-e meg valamelyik könyvtárbejegyzés (például egy állomány hossza) előző állapotához képest (feltéve, hogy feljegyeztük és fájlban eltettük az előző állapot leírását);

— ha gyorsan ellenőrizni akarjuk: nem más változat programjai jelennek-e meg azonos néven, vagy megfordítva, nem valami átnevezett, régről ismert program bukkan-e fel esetleg más köntösben (hiszen sem a név, sem a dátum nem jelent bizonyosságot a belső tartalmat illetően).

Az utóbbi két esetben rendszerint már nagyobb hozzákészülődés szükségeltetik: „kézzel vezérelt" egyedi összehasonlítások helyett célszerű felkészülnünk csoportos összehasonlításokra is. Egész könyvtárak összehasonlításához tálcán hozza a kész megoldást a Norton Commander beépített „Compare directories" utasítása. De speciális kívánalmaknak megfelelően magunk is készíthetünk testre szabott batch fájlt a COMP.COM utasítás felhasználásával. Háziszabványszerű névválasztási konvenciókat is kidolgozhatunk magunknak a COMP.COM dzsókerkezelő lehe-

tőségeivel összhangban. (Vigyázat! Ha mindkét argumentumban dzsókerkaraktereket alkalmazunk, nem mindegy, hogy milyen sorrendben adjuk meg az argumentumokat: a második argumentum szerint végrehajtott helyettesítések az első argumentum alapján létrejött karaktermintákat viszik tovább.)

Ne a program döntsön!

Ami igazán bosszantó lehet az ilyenféle, hagyományos ellenőrző-hasonlító programokban, az nem is az, amit nem tudnak, hanem amit nem hajlandók megcsinálni. Szinte kivétel nélkül hosszhasonlítással kezdik tevékenységüket, és ha itt eltérést tapasztalnak, hozzá sem fognak az érdemi munkához. Pedig, mondjuk, rákérdezhetnének, hogy eltérő hosszuk ellenére is ragaszkodunk-e a parancs végrehajtásához. Már engedelmet, de ne a program döntse el, hogy van-e értelme egyáltalán az összehasonlításnak! Hogy csak néhányat említsek az érdekesebb esetek közül, amikor teljesen indokolt lenne különböző hosszúságú állományok összehasonlítása:

— Keletkezhetnek különböző hosszúságú, bár effektív, működő részükben tökéletesen megegyező programok, például transzformációk és a megfelelő inverz transzformációk alkalmazása következtében. Amikor például az LZEXE programmal tömörítünk, majd az UNLZEXE programmal visszaállítunk valamilyen .EXE állományt, hosszuk különböző lehet.

— Egyes programoknak azonos változatszámú, mégis eltérő hosszúságú mutációi vannak forgalomban. (Az LHARC 1.14-es változatának kétféle mutációját ismerem.) Valóban érdemlegesen eltérnek egymástól, vagy az eltérés például csak a beépített információs szövegek különbségéből adódik?

— Vírusok, de vírusok elleni „védőoltások" is megnyújthatják a programokat. Sok esetben hasznos lehetne, ha látnánk, hogy valóságos támadásról van-e szó.

— A könyvtárfájlok összehasonlításakor ritka kivételként fordulhat csak elő, hogy a hosszuk megegyezzék. Éppen arra lenne szükségünk, hogy ne csupán az eltérés tényét konstatáljuk, hanem részletezve lássuk, miben áll az eltérés.

Egyes esetekben némi ügyeskedéssel túljuthatunk a nehézségeken, ha tudjuk, hogy a FIND.EXE „szűrőutasítás" opciótól függően vagy átereszti vagy elnyeli (/v) a feltételnek eleget tévő sorokat. Valójában azonban itt már csődöt mond a hagyományos „ellenőrző-hasonlító" programok tudománya. A minőségileg magasabb szint ott kezdődik, ahol a programok nem elégednek meg annyival, hogy a pozíciószámban egymásnak pontosan megfelelő karaktereket hasonlítják össze. Hiszen mi történik mondjuk egy szöveges állománnyal egyetlen szó vagy akár csak egyetlen karakter beszúrása vagy törlése után? Az összes többi karakter menthetetlenül elcsúszik vagy egyik, vagy másik irányba.

Milyen siralmas eredményt adhat ezek után egy „hagyományos" összehasonlító program, ha a változás már valahol a szöveg elején bekövetkezik? Kisiklik menthetetlenül, hiszen az első eltérés észlelése után elcsúsznak egymáshoz képest az egymásnak megfelelő karakterek. A mechanikus, fixen pozicionált összehasonlítás itt nem segít, hiszen sorban egymás után hibát jeleznének az összes további összehasonlított karakterek. Hiába segítenénk tehát a hosszkülönbségből adódó akadályon keresztül a programot, képtelen lenne felismerni a legnyilvánvalóbb hasonlóságot is.

Az „objektum" fogalma

Tudomásom szerint a Turbo Pascal programozói készítettek először olyan programot, amely már nem butácska módon, vakon botladozott az akadályok között, hanem tudatosan gyűjtötte és hasznosította az információkat a probléma értelmes megoldásához. A DIFF segédprogram Pascal nyelvű listáját is közzétették 1985-ben, a TP 3.0 verziójának a megjelenésekor. A felhasznált algoritmus *Dave Cortesi* műve, az elméleti alapvetés azonban régebbről származik: 1978-ban publikálta ez irányú eredményeit a CACM-ben *Paul Heckel* („A Technique for Isolating Differences between Files").

A Heckel-féle megközelítés lényege, hogy a részleteiben összehasonlítandó fájlokon belül bevezeti az „objektum" fogalmát, szisztematikusan összegyűjti a rájuk jellemző fontosabb ismérveket, majd ennek alapján dönti el, hogy melyik objektum melyik másiknak feleltethető meg. (Alapvető objektumnak szöveges fájlok esetében a sorokat szokták tekinteni.) Minden objektumot egy-egy szimbólum képvisel egy dinamikusan kezelt szimbólumtáblázatban. Ennek alapján történik az összehasonlítás és a értékelés, részint az egyes objektumokra jellemző ismérvek, részint az objektumok között fennálló sorrendiség figyelembevételével.

Látjuk tehát, hogy a mechanikus szempontok jelentősége háttérbe szorul a feladat megoldásában. Leginkább az alakfelismerés problémakörével mutat hasonlóságot a megoldandó feladat, azzal a csavarral nehezítve, hogy itt nem létezik eleve a felismerendő objektumoknak valamilyen véges halmaza. A kiválasztandó minták mindig módosulnak, sőt azt sem tudhatjuk előre, melyiknek lesz megfelelője a másik fájlban, és melyek maradnak facéran. (A karakterfelismerésben például mindig tudjuk, hogy ha nem valamilyen piszokról van szó, akkor a karakterek adott véges halmazában kell lennie a felismerendő objektumnak.)

A DIFF program persze sokkal jobb eredményeket ért el, ha nem tetszőleges szöveges állományok összehasonlítására alkalmazták, hanem ki lehetett használni tájékozódásul például a Pascal kulcsszavait mint jellemző ismérveket. Úttörő szerepét a DIFF elfogadható módon betöltötte, forráslisták összehasonlítására ma is jól használható. Kimeneti listaként a „beszúrt" és „kihagyott" sorok felsorolását kapjuk meg mindkét szövegre. Részleges egyezéseket nem mutat ki — ilyenkor mindkét változat „beszúrt" szövegsorként jelentkezik.

Egy újabb gyöngyszem

Hasonló elvek alapján készült el 1988-ban egy újabb okos kis összehasonlító program. A program (és az algoritmus) szerzője *M. J. Mefford*, a PC Magazine ismert munkatársa, aki már sok „gyöngyszemmel" örvendeztette meg a felhasználókat.

A DIFF-fel ellentétben a COMPARE többféle szintű objektumokat ismer: a sorokat, a szavakat és a szavakon belül az összefüggő karakterláncokat. Ha a sorok összehasonlításával problémái vannak, alacsonyabb szintre száll le, úgy folytatja a munkáját. Ez a megközelítés gyakran beválik, sőt finomabb és tetszetősebb megoldásokra is vezet: a képernyőn szemléletesen mutatja is az egymásnak megfeleltetett szövegrészeket, különböző színekkel, illetve inverz kiemeléssel jelezve az eltéréseket. Időnként azonban elég könnyen megzavarodik a rendszer, aminek tapasztalataim szerint kettős oka van. A fő ok az, hogy mindig az első szöveg alapján választ mintát, ahhoz keresi a másik szövegben a megfeleltetést. Ha tehát a minta valamilyen kitörölt szövegrészből származik, a találat eleve csak hibás lehet. A másik hibaforrás szorosan összefügg ezzel: ha már áttért az alacsonyabb szintre, könnyen becsapják a

részleges eredmények, és azt hiszi, jó úton jár.

A COMPARE erőssége, hogy nemcsak ASCII fájlokat, hanem tetszőleges bináris fájlokat tud (hexa-alakban) összehasonlítani, egyszerre jelenítve meg a hexa-számokat és karakteres megfelelőjüket. (A szöveges fájloknak is kérhető az összehasonlítása ilyen formában.) A COMP makacssága tehát, hogy nem hajlandó különböző hosszúságú állományokat összehasonlítani, ezzel leküzdhető. A DIFF-fel szemben viszont előnye is, hátránya is a képernyő-orientáltság: a képernyőn szemléletesen mutatott eltérések az egymásnak megfeleltetett szövegrészek között nem vihetők ki a nyomtatóra.

Érdemes megfigyelni, hogy a hasonlóság vizsgálata más-más eredményre vezet attól függően, hogy milyen sorrendben adjuk meg az összehasonlítandó állományok nevét. A „hasonlóság" Mefford-féle algoritmikus fogalma tehát nem szimmetrikus, ami némileg zavarja érzékeny matematikusi lelkületünket.

Egészében véve (és különösen méretéhez képest) igen ügyes és sokoldalú program a COMPARE, és tegyük hozzá: szerkezete is kristálytiszta. Kár, hogy első változatának 1988-ban történt nyilvánosságra hozatala óta — tudomásom szerint legalábbis — nem jelentek meg további változatai. Egyszerűbb esetekben kiválóan alkalmazható, egészen odáig, míg valahol bele nem zavarodik, mert attól fogva képtelen kigabalyodni a hibás egybevetések hálójából. Egy-egy véletlenül megegyező szót vagy kisebb karakterláncot találatnak tekint, és rendületlenül gyártja a hibás megfeleltetéseket. Feltétlenül érdemes lenne valakinek — ha már a szerző nem teszi — továbbfejleszteni a program összehasonlító algoritmusát.

Három hetente új változat

Az Alaplap 1991. januári melléklete érdekes, sokat tudó fájlhasonlító programra hívta föl figyelmünket. A Magyarországra nemrég eljutott Delta nevű programrendszer (ez a minősítés jobban megilleti) valóban megérdemli, hogy felfigyeljünk rá. Bár a jelenleg elérhető verzió még távolról sem tekinthető tökéletesnek, újszerű és látványos szolgáltatásaival olyasmit nyújt, ami sok szempontból túlhaladja az e műfajban ismert programok tudományát. Nem csoda, hogy első nyilvános megjelenése, vagyis 1989 áprilisa óta komoly felhasználói tábora alakult ki az Egyesült Államokban.

A regisztrált felhasználók az OPENetwork hálózat éjjel-nappal működő BBS szolgálatán keresztül adhatják át tapasztalataikat és kívánságaikat a fejlesztőknek, kérhetnek tőlük segítséget, és innen hívhatják le a program legújabb változatát. A felhasználók kívánságainak figyelembevételével folyik a program továbbfejlesztése. (Átlagban 3 hetente jelenik meg újabb változat, ez a tempó azonban idővel feltehetően lelassul.) A fejlesztés részben a felhasználók kényelmét és biztonságát, részben a program működésének gyorsítását és helyfoglalásának mérséklését szolgálja, ami egyébként rá is fér.

Tíz szemmel figyel

Mefford programjával szemben a Delta összehasonlítási algoritmusa a DIFF-hez hasonlóan a sorok összehasonlítására épít, ezen belül azonban kívánságra figyelembe veszi a részleteket: színes képernyőn szépen mutatja a törölt, illetve beszúrt szavakat, karaktereket. A szövegrészek egymáshoz illesztése minden esetben a sorok egybevetése alapján történik. Ez általában jó, mert így sokkal ritkábban zavarodik meg, mint a COMPARE, de például bajba kerül, ha különböző hosszúságúra tördelt szövegeket kell összehasonlítania. (Mefford ilyenkor áttér az alacsonyabb egységek szintjére, és a szavak szintjén sikeresen folytatja az összehasonlítást.)

Ügyesen oldja meg a Delta (sok munkafájl nyitva tartásával) a pszeudoszimultán összehasonlítást. Egyszerre tíz „szemmel" figyel előre és hátra — főleg ez a magyarázata viszonylag gyors és kevésszer hibázó megfeleltetéseinek.

Igen hasznos és rendkívül gyors szolgáltatása a Deltának teljes könyvtárak összehasonlítása. Az eredményül kapott, jól áttekinthető párhuzamos lista rögtön ki is nyomtatható, ami dokumentumnak sem utolsó. Képernyőre kérhető (de csak PrScr-rel nyomtatható) az azonos nevű, de hosszukban és/vagy tartalmukban különböző fájlok listája.

Könyvtárak összehasonlításakor válogatás nélkül elbánik bináris fájlokkal is, szigorúan a hagyományos módon kezelve őket, egyébként azonban csak ASCII fájlok kezelésére van fölkészülve. Ha igaz, lennie kell viszont analóg programnak bináris fájlokra is — legalábbis erre utal a program induló menüjében az „ASCII edition" megkülönböztetés.

Amikor áttérünk a könyvtárak szintjéről a fájlok összehasonlításának a szintjére, a függőlegesen vagy vízszintesen osztott képernyővel egymás mellett vagy egymás alatt látjuk az összehasonlított állományok egymásnak megfelelő részeit. (Ezt nevezi a Delta a „sztereó megjelenítés" két módjának.) Gombnyomás biztosítja az átmenetet a kétféle sztereó megjelenítés között. A két félképernyő a kurzorbillentyűkkel szinkronban mozgatható.

Beépített szövegszerkesztője kissé nehézkes, de hát nem is ez a Delta elsődleges rendeltetése. (Őszintén szólva szívesebben venném, ha inkább a jól bevált szövegszerkesztők látnák el a Delta szolgáltatásait is.) Egyidejűleg csak az egyik szöveg szerkesztését teszi lehetővé, de teljes sorokból álló egységek formájában átemel szövegrészeket a másik állományból. Sok blokkműveletet ismer, sőt kijelölt blokkokat tesz ki megadott nevű fájlba. A képernyőn nem zavarják a magyar ékezetes karakterek, nyomtatásnál azonban sajnos lecseréli őket kérdőjelekre.

Számos opciót kínál föl a program, a részletek megjelenítésétől kezdve a szerkesztőgombok átdefiniálásán és az eltérések kimentésén át az egymásutáni szóközök és látszólagos szóközök (00, FF) ismétlődésének kiszűréséig. Kár, hogy másutt viszont opcionálisan sem tűr beleszólást a működésébe. Nem ártana például megengedni — a felhasználó kívánságától függően — a Pascal értelmében vett „átlátszó" karakterek kezelését. Igaz, ez lassíthatná működését, de sok hibaforrást megszüntetne.

Hasznos segítőtárs

Az opciók kezelése is sajátos kissé: előfordul, hogy önkényesen semmibe veszi az előírásokat. Egy jól nevelt programtól szokatlan, hogy néha utólag derül csak ki: hiába kértük az összehasonlításban a beépített „mesterséges intelligencia" adta lehetőségek felhasználását. Ha ugyanis helyzetértékelése alapján indokoltnak tartja, se szó, se beszéd, fogja magát és átkapcsol mesterségesen buta üzemmódba. Szerinte ti. gyökeresen eltérnek egymástól az összehasonlított állományok, ami persze nem minden esetben igaz.

Mindez azonban csak részletkérdés, ami a program sokoldalú használhatóságát alapvetően nem befolyásolja. Kimondottan hasznos segítőtársra találhatnak benne mindazok, akiknek munkájába rutinszerűen beletartozik a szövegeltérések figyelése, különböző fázisban lévő szövegek egybevetése, vagy éppen az adatrögzítés (például ügyvédek, újságírók, olvasószerkesztők, kisvállalkozók).

Vargha Dénes

Szövegkonvertálás

Sok felhasználónak okoz gondot a többféle magyar ékezetes karakterkészlet közötti konvertálás. Teljes megoldást nyilvánvalóan csak a magyar karakterkészlet szabványosítása jelentene, de erre egyre kevesebb esély látszik. Egyelőre marad tehát a különféle konvertálóprogramok használata. Most ezekből kettőt ismertetünk. Az egyik a 8-bites karakterkészletek közötti átalakítást végzi, a másik a WordStar fájlokat 8-bites ASCII fájlokká konvertálja.

A CASCII.C programot két fájl megadásával kell indítani. Az első az input-, a második az output-fájl. Formátumára a program a fájl kiterjesztéséből következtet. Jelenleg három kódtáblát támogat:

1. CWI ASCII

A Computerworld-Számítástechnika által kezdeményezett kódkészlet (.CWI kiterjesztésű fájl).

2. Ventura ASCII

A Ventura Publisher által használt kódtábla (.TXT kiterjesztésű fájl).

3. BME ASCII

A Budapesti Műszaki Egyetemen használatos kódtábla (.BME kiterjesztésű fájl).

A további bővítés igen egyszerű: a program elején lévő konverziós táblázat egy újabb oszlopát kell létrehozni az új kódokból. Ezután a második indexet 3-ról 4-re kell átírni. A kiterjesztést a kiterjesztéseket tartalmazó táblázatba kell beírni. A program Turbo C 4.0 vagy C++ 1.0 fordítókkal fordítható újra.

Működését tekintve a program három részre tagolódik. A főprogram feladata a parancssori paraméterek és a formátum ellenőrzése, a használati utasítás kiírása, a fájlok megnyitása és lezárása. A tényleges konverziót a Convert( ) eljárás végzi. Karakterenként beolvassa az input fájlt, majd a konverziós tábla input fájlformátumának megfelelő oszlopával hasonlítja össze. Ha egyezést talál, akkor ugyanabban a sorban, az output formátumnak megfelelő oszlopban álló karaktert írja ki. Ha nem talál, akkor a karaktert változtatás nélkül teszi az outputfájlba. A program másik rutinjának, a Strcode()-nak, a kiterjesztés felismerésénél van jelentősége. Az eljárás képes megkeresni egy sztringet vagy annak rövidítését egy tetszőleges táblázatban. Erre gyakran van szükség, s ezért ez az eljárás számos példaprogramunkban előfordult már.

A WordStar még néhány évvel ezelőtt is a legelterjedtebb editornak számított. Népszerűsége mostanra ugyan csökkent, de számos editor (pl. a Turbo-C editora) valósít meg WordStar-kompatibilis funkciókat. A WordStar legnagyobb hibája, hogy nem alkalmas 8-bites ASCII-fájlok létrehozására. A WordStar 2.0 verziója még csak a 7-bites kódokat támogatta, a legfelső bitet más célra használták. A későbbi verziók a 8-bites kódok tárolására hárombájtos szekvenciát használnak. Ezt a formátumot az előző programmal nem lehet konvertálni, ezért önálló program írására került sor.

A WordStar fájlokból 8-bites ASCII állományokat konvertáló programot az input- és az output-fájl megadásával kell indítani. A program két részre tagolódik: a főprogram feladata a paraméterek ellenőrzése, a használati utasítás kiírása, a fájlok megnyitása és lezárása. Ez a rész az előző program főprogramjának egyszerűsített változata. A konverziót a Convert( ) eljárás végzi. Ez a 1BH xx 1CH szekvenciákat xx-szel helyettesíti, és az összes többi karakter legfelső bitjét törli.

Pintér Gábor

Kód-gettó

A Microsoft május végén kétnapos konferenciát rendezett Prágában a kelet-európai szoftverfejlesztőknek. A fő attrakció természetesen az új Windows-környezet bemutatása volt, de számos más érdekes téma is terítékre került, például a kelet-európai nyelvek betűinek korrekt használata a számítógépeken.

Tisztelt Microsoft! Bármennyire is respektáljuk erőfeszítéseiket a fránya kontinentális európai nyelvek ékezetes betűit tartalmazó kódkiosztás megoldására, sem a 852-es (Latin-2) kódtáblával, sem a most bemutatott, ANSI szabványon alapuló kelet-európai Windows 3.1 kódkészlettel nem tudunk megbékélni.

Tegezésre kell átváltanom, hogy egy régi magyar sláger refrénjével tudjak rimánkodni: „Hogyan mondjam el neked?" Mármint azt, hogy az európai kis nyelvek nincsenek egymással úgy összefonódva, mint a nyugat-európai világnyelvek. Bele kellett volna kicsit nézni az érintett országok szövegszerkesztőibe, adatbázisaiba, irodalmi és egyéb kiadványaiba, hogy milyen arányban használják például Magyarországon a magyar nyelvű szövegekben az albán, a román, a cseh, a szlovák, a lengyel vagy a szerbhorvát speciális karaktereket. Azokat, amelyek most egyazon kódtáblában szerepelnek.

Tisztelt Microsoft! Nem földrajzi, nem nemzetiségi, nem politikai kérdésről van szó, hanem a nyelvi realitások józan belátásáról. Ugyanúgy értelmetlenség lenne egy másik összevont kódtáblát kreálni a holland, a finn meg a török nyelv speciális ékezetes betűire — és abból is kihagyni a nagy világnyelveket. Inkább az ASCII kódtábla szabványában kellett volna szabad helyet hagyni a nemzeti karaktereknek — ha már az elején elmulasztották, akkor legalább utólag —, mégpedig annyit, hogy azon bármelyik európai nyelv egyéni betűi elférjenek. De mindig csak egy renitens nyelvé! Mellette pedig ott maradna — minden országban — ugyanaz a nemzetközi kódkészlet! Ehelyett továbbra is elkészül időnként egy-egy karakter-kód-gettó, amit használni nem tudunk. Amitől legfeljebb sírni tudnánk. Vagy már csak röhögni?

Faklen Pál

FoxPro 2.0
Nem slowFoxot táncolva

A PC adatbázis-kezelők vetélkedője folytatódik. A hagyományokon alapuló dBASE IV adatbázis-kezelőnek már a FoxPro 1.0 változata is komoly konkurenciát jelentett, elsősorban sebessége és kellemes felhasználói felülete miatt. Most pedig a FoxPro 2.0 nemcsak megőrizte előnyeit, hanem néhány igazán jó fejlesztéssel is előrukkolt.

Menüzgetés

A FoxPro 2.0 az 1.0 verziótól megszokott módon jelentkezik be. A teljes rendszer SAA menükből vezérelhető, szinte nem is kell a Command ablak, egérvezérelt, és természetesen számtalan ablak nyitható a különböző műveletekhez. A FoxPro 2.0 támogatja a Windowsból ismert gombok, dobozok, dialógusok használatát. Ezzel a FoxPro 2.0 ún. karakterizált grafikus felhasználói felületet hozott létre (CGUI).

A Help-rendszer hívható a System menüből, a Command ablakból és az F1 leütésével. A System ablakból, menükből, dialógokból és Edit vagy Command ablakból kiválasztott szöveghez az Alt+F1 leütésének hatására azonnal a kívánt segítőszöveg jelenik meg. A Help-részek alján felsorolt kapcsolódó fogalmak kiválasztása újszerűen a See Also kiválasztásakor megjelenő Popup menüből történik. Ennek segítségével tudunk vissza is térhetünk a kiinduló fogalomhoz.

A System menüből is hívható a Help-rendszer, és saját billentyűmakrókat definiálhatunk. A menüből sok segédprogram közvetlenül elérhető, például a Filer, Calculator, Calendar/Diary, Special Characters, ASCII Chart, Capture. A fájlmenedzser felajánlja a meghajtók, könyvtárak és fájlok kezeléséhez szükséges műveleteket: Copy, Move, Delete, Rename, Attr, Edit, Sort, Tree, Find, Chdir, Mkdir, Size. Így pár gombnyomással akár alkönyvtár szinten mozgathatunk, másolhatunk, törölhetünk. Rendkívül praktikus a Capture menüpont, amellyel a teljes képernyőn (ablakoktól függetlenül) kijelölhetünk egy képernyőterületet, azt egy átmeneti tárolóba (az Edit/Copy funkciónak megfelelően) elmenthetjük, majd később egy másik tetszőleges helyre (például egy ablakba) ismét beilleszthetjük (Edit/Paste).

A File menü is gazdagabb. Lehetőség van a már megszokott Database, Program, File, Index, Report és Label típusok mellett újabbak (mint például Screen, Menu, Querz és Project) létrehozására, megnyitására. A Report, Label, Screen, Menu, Query és Project aktivizálásakor az adott állománnyal végzett műveletek támogatására a képernyőn feltűnik a típusnak megfelelő menüpont is. A Quick Report, Quick Label, Quick Screen és Quick Menu kiválasztásával a rendszer önállóan elkészíti a szükséges állományokat. Ilyen lehetőségek sokasága alapján akár a kezdő felhasználók is gyorsan profi szintűnek látszó alkalmazásokat tudnak készíteni.

Az Edit menü tartalmazza a FoxPro standard editáló technikáját (Undo/Redo, Cut/Copy/Paste, Clear és Select), amely a textboxokkal és dialógokkal segíti az editálást. A Preferences menüpontban texteditáláskor például beállíthatjuk a Wrap Words, Cheked és Uncheked funkciókat, de ha az Expression Builder aktív, akkor kifejezéslisták állíthatók össze a segítségével matematikai, string-, logikai, dátumfüggvények redőnymenüjéből, mezőnevek és változólisták felhasználásával.

„Rushmore"-technikával

Az adatbázis-állományok nyilvántartásának legnagyobb újdonsága a rushmore-technika. Az egy időben megnyitható adatállományok száma még korlátozott (maradt a 25 munkaterület), de ez valójában elvi határ, a gyakorlatban nem igazán korlátozza a használatot. Az indexelés területén viszont korszakalkotóan újat hozott a FoxPro. Már a dBase IV is próbálkozott az egyedi indexek mellett összetett indexek alkalmazásával, de ez valójában csak a 47 szempont szerinti indexek egy meghatározott sorrendje. A FoxPro is kétféle (s azon belül is kétféle) indexet használ. Megmaradt az előző verzióból ismert .IDX egyedi indexforma, de a rendszer elsősorban a .CDX (Compound index) összetett index használatát támogatja. Az .IDX egyedi indexet valójában a kompatibilitás érdekében használja, ezért létezik compact- és non-compact-változata is. A non-compact-index megegyezik a FoxPro előző verziójának, a FoxBase+-nak az indexformájával, ezért azokat a FoxPro 2.0 eredeti formájukban használja, tárolja. Ha azonban nem használják előző verziók az indexfájlt, érdemesebb a compact formát igénybe venni, mert tömörebb és gyorsabb.

Indexek létrehozásakor a rendszer elsősorban az összetett indexformát ajánlja, amelynek szintén két típusa van. A structural index felveszi az adatállomány nevét, és automatikusan vele együtt kezelhető (a Use adatállomány hatására automatikusan megnyitja és aktualizálja a rendszer), míg a non-structural indexnek mi adhatunk nevet, de akkor megnyitásáról és aktualizálásáról nekünk kell gondoskodni. Az összetett index lényege az, hogy az egyes indexeket rekordokként kezeli, ezért például sorrendjük sem lényeges. Ezenkívül egy fájlnak számít a DOS File Handle szempontjából, s így az indexek száma gyakorlatilag végtelen

Majdnem Windows-szinten:
- egérvezérlés, SAA menük,
- kifinomult ablaktechnika.
- Context-sensitive Help

SQL elemek:
- SELECT, CREATE TABLE,
- INSERT-optimalizálás

Screen, menütervezés
Projectkészítés
Rushmore-technológia
Trace, Debug funkció
.EXE állományok készítése

lehet. A compact .IDX és a .CDX indexeket a rendszer tömörített formában tárolja (kb. egyhatod lemezszükséglet), majd a memóriában szükség szerint kicsomagolja. Azon műveleteknél, amelyeknél elég az indexállományok használata, megtakarítható a lemezhez fordulás ideje, s így akár ezerszeresen gyorsabbak lehetnek a hagyományoshoz képest. A rushmore-technika optimalizáló eljárása elsősorban azoknál a FOR-parancsopcióknál és FILTER-műveleteknél gyorsít jelentősen, ahol a feltételben szereplő mező szerint létezik index.

SQL

A dBASE IV-hez hasonlóan a FoxPro 2.0 is használ már SQL-parancsokat. Lehetőség van adatbázis vagy tábla létrehozására, rekordok hozzáfűzésére és lekérdezésére is, mindezt a FoxPro 2.0 komfortjával. Az RQBE- (relational query by example) ablakban lehetőség van relációkkal összekapcsolt állományok komplex lekérdezésére egyszerű módszerekkel. A FoxPro ezeket a lekérdezéseket is optimalizálja a rushmore-technika alapján. Az összeállított lekérdezés lemezre rögzíthető, sőt programként tárolható, lefordítható és futtatható is. Szenzációt sejtet a Create Table-funkciónál megmutatott, de egyelőre nem választható Picture mezőtípus. Lehet, hogy ez már a grafikus felület alkalmazását készíti elő? Kíváncsian várjuk a folytatást.

A FoxPro 2.0-ban az előző verziók gyenge pontjaként számon tartott képernyőtervezőt teljesen újraprogramozták, a rendszerbe illesztették. Ez a rendszer integrálja az SAA menüket, támogatja a mouse-vezérlést. A Screen menü segítségével gyorsan esztétikus képernyőt tervezhetünk (hát még a Quick Screennel), majd lemezre menthetjük. Az elkészült állományt programba fűzhetjük, vagy project készítéséhez használhatjuk. A FoxPro 2.0 képernyőállománya nem kompatibilis a többi adatbázis-kezelőével, de támogatja a külső .FMT állományok használatát. Ezek a programba illeszthetők, bár mint screen állományok, nem módosíthatók.

Az új menügenerátor használatával gyerekjáték bonyolult pulldown menüstruktúrák létrehozása. Először meg kell adni a főmenü pontjait, majd azokhoz műveletet (Command, Pad Name/Bar#, Submenu, Proc) kapcsolni, és opciókkal is ellátni. Az opcióknál megadható például, hogy az adott menüpontot milyen feltételek esetén lehet választani (hasonló módon, mint az Expression Builder). Ha egy menüszint elkészült, jöhet a következő ugyanígy. A végén lehetőség van a menü tesztelésére, a megadott parancsok, utasítások tényleges végrehajtása nélkül. Az elkészült menüt a képernyőterv-állományokhoz hasonlóan használhatjuk fel.

A Report- és Label-generátort már az előző verzióknál is jól használhattuk, de azért itt is sikerült újítani. Egyrészt a Quick Report és Quick Label használata nagyon kellemes, de leginkább az elkészített Report és Label Preview funkciója hasznos. Megtekintése alapján ugyanis eldönthetjük, tetszik-e munkánk eredménye, vagy van még javítanivaló, s nem kell közben kilépni a generálóból.

A megtárgyalt object típusú Report, Label, Screen, Menu és Query mellett természetesen lehet normál módon is programot írni, Text/Program Editor ablakban módosítani, és parancsot adni a Command ablakban. (De minek?)

Csemege haladók számára

Az elkészült programok ritkán hibátlanok. A hibák felderítésében és kijavításában segít a Trace és a Debug ablak. Lehetőségünk van a forrásszövegen követni a végrehajtást, változókat lekérdezni, javítani. Mindezekre a koronát a Project Manager teszi fel. Miután elkészültek a Database, Index, Program, Report, Label, Menu, Screen, Query és egyéb állományok, egy projectbe fogja az összetartozó objektumokat. Ezt project adat- és memo-állományban rögzíti. A Project menü és ablak használatával információkat és statisztikákat kérhetünk az objektumokról, az állományokból egységes programot szerkeszthetünk. Ha hiányzik egy állomány (és nem találja sehol), vagy más hibát talál, azt jelzi, és lehetőséget nyújt a javításra.

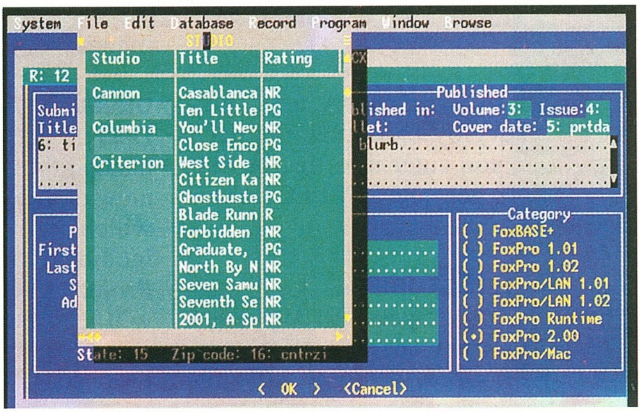
Már nem csak a Clipperrel lehet .EXE állományokat fordítani. A FoxPro 2.0-ban a fordításnak több szintjét különböztetjük meg. Egyrészt létezik a már megszokott közbülső állomány, amit csak a FoxPro programmal lehet indítani, de már futtatható .EXE állományt is lehet készíteni. A futtatható állomány mellett itt is szükséges overlayként segédállomány, de nem a FOXPRO.EXE. A Clippertől eltérően a rendszer nem fordítódik hozzá minden egyes .EXE állományhoz, hanem több futtatható állomány is használhatja ezeket az overlayket.

A FoxPro gyors és gazdaságos memóriakezelése mellett is előfordulhat, hogy nagyobb alkalmazásoknál nem elég a memória. Ezért a jobb gépi adottságok kihasználására elkészítették a FoxPro extended (X-szel jelzett) verzióját is, amely olyan PC 386-os és 486-os gépeken fut, amelyek legalább 2 MB memóriával rendelkeznek. Ez a bővített verzió mindkét processzortípusnál a teljes operatív tárat közvetlenül címzi, a memóriafoglalást optimalizálja.

A FoxPro 2.0 megengedi a rendszer bővítését. Ehhez a gyakorlott programozóknak az API (application program interface) vezérszó alatt programok, definíciók, funkciók sora található, amelyeket bővíteni lehet C és assembly nyelven írt programokkal.

Mindezek alapján a FoxPro 2.0 a PC-re készült relációs adatbázis-kezelők közül jelenleg a legjobb. A legkomfortosabb, leggyorsabb adatbázis-kezelő címet sokáig nem vitathatja el tőle senki.

Csiki András



Zephyr

Adatbázis — igényeseknek is

Közkedvelt számítógépes alkalmazás az adatbázis-kezelés, mert nagy adattömegek sokoldalú szervezését, elérését és archiválását teszi lehetővé. Az adatbázis-kezelő rendszerekkel gyorsan megkereshetünk például egy árutételt, vagy az adatállomány alkotóelemeit változatos szempontok szerint rendezhetjük, csoportosíthatjuk. Az adatbázis-kezelők családja nemrég egy igen jó tulajdonságokat felmutató taggal gyarapodott. Ő a Zephyr.

A lehetséges felhasználási területek sokrétűsége azt eredményezte, hogy a fejlesztők áttekinthetetlen mennyiségben kínálják a különböző adatbázis-kezelőket. Létrehoztak például speciális adatbázis-kezelőket kifejezetten címjegyzékek vagy szakirodalom céljaira. Abban az esetben azonban, ha otthon a háziasszony a receptjeit akarja gépre vinni, férje pedig a számlakivonatokat akarja nyilvántartani, ezeket a „célorientált" adatbázis-kezelőket át kell alakítani. Léteznek azonban univerzális adatbázis-kezelők is, például az Ashton-Tate terméke, a dBASE vagy a Fox Software cég Foxbase programja.

Az adatbázis-programozók számára olyan programozási segédeszközöket is kifejlesztettek, amelyekkel igen jó teljesítmény érhető el az adatbázisok létrehozásánál. Ilyen programozási segédlettel készült a Zephyr program, amely a dBASE-hez és a Foxbase-hez hasonlóan a legkülönbözőbb célokra használható adatbázis-kezelő rendszer.

A programot két, 360 kilobájtos floppyra vitték fel. Ezeken találhatók az önkicsomagoló állományok, kicsomagolás előtt pontosan leírva azokat a lépéseket, amelyek a sikeres installáláshoz szükségesek. Ha követjük ezeket az utasításokat, azok a rendszerelemek, amelyek feltétlen szükségesek az egyes programrészek működéséhez, gyorsan rákerülnek a merevlemezre.

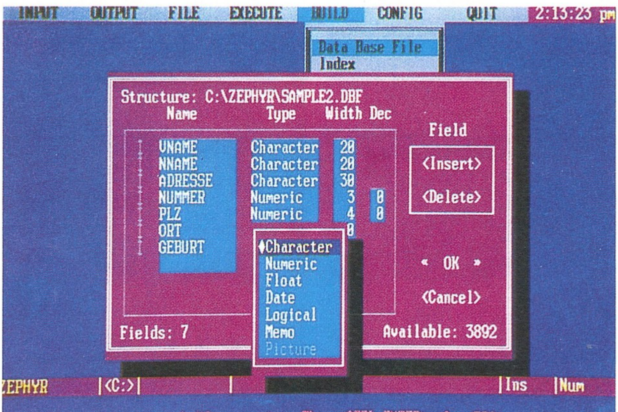
A program egyes funkcióinak megismerését egy angol nyelvű, 46 oldalas kézikönyv segíti, amely a floppyn található. Ha kinyomtatjuk, tájékoztat a Zephyr lehetőségeiről. Az egyes utasításokkal kapcsolatosan csak bevezető jellegű információkhoz jutunk, hiszen a FoxPro adatbázis-kezelő nyelv alapján igen sok művelet áll rendelkezésünkre.

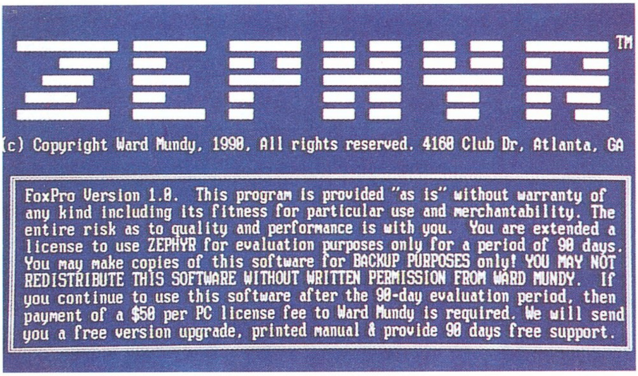
Szükségünk lehet természetesen az utasítások bővebb ismertetésére is, amit az on-line helpben találunk meg. Ez gyakorlatilag egy saját Zephyr-adatbázis, az F1 funkcióbillentyű hatására jelennek meg egyes részei a képernyőn.

A Zephyr alkalmazásával nagyon egyszerűvé válik az adatbázisok felépítése és kezelése, ugyanis igen jól sikerült felhasználói felülettel rendelkezik, jól vezeti munkája során a vele dolgozókat. Menükből és almenükből aktivizálhatók az egyes műveletek. Az állományok betöltése, mentése és az ehhez hasonló műveletek mellett külön menüpont szolgál az adatbázis-struktúra definiálására. Működési módja nagyon hasonló a dBASE-éhez. Először az egyes adatmezőket kell megadnunk, ez után következhet az adatbevitel, az így bevitt adatokkal pedig műveleteket végezhetünk.

A mezők definiálásakor is hasonlóképpen dolgozhatunk, mint a dBASE-nél. Különböző mezőtípusokat választhatunk, lehetnek például dátumtípusú, szöveges és numerikus mezőink. Kitüntetett adattípus az úgynevezett „memo"-mező, ez ugyanis saját állományt hoz létre az adatbázisban. Ebbe az állományba tetszőleges hosszúságú szövegek írhatók be a Zephyr beépített szövegszerkesztőjével. A kiegészítő információ tehát bármekkora lehet. Irodalomfigyelésnél például a Zephyr „memo"-mezői lehetővé teszik, hogy problémamentesen kezeljünk teljes cikkeket is.

Számításainkhoz pedig olyan mezőket definiálhatunk, amelyekben automatikusan lefutnak az adott matematikai műveletek akár előre megadott, akár lekért adatokkal.





Mivel minden bevitel a menük alapján, jól definiált inputmezőkön keresztül történik, valamint megvalósult a teljes körű egértámogatás SAA alapon, az egyes műveletek kiszolgálása nagyon egyszerű, könnyen megtanulható, hiszen minden ugyanarra az elvre épült.

Az adatbázis-struktúra felépítése mellett úgynevezett indexmezőket is létrehozhatunk. Ezekbe olyan adatbázismező kerülhet, amelynek alapján rendezni akarjuk állományunkat, illetve amelyre keresni akarunk majd az adatbázisban. Indexállományokat is létrehozhatunk az egyes indexmezőkből: ez az összekapcsolás a FoxPro program utasításaival történhet.

Adatbázisonként maximum 25 indexmezőt enged meg a Zephyr, ezek közül választhatjuk ki azt, amelynek alapján rendezni vagy keresni akarunk. Indexállomány váltásakor azonnal az új rendezettséggel dolgozhatunk. Mivel a Zephyrben egyidejűleg 25 adatbázist nyithatunk meg és dolgozhatunk fel, gond nélkül definiálhatunk egyidejűleg több adatbázist is. Ezt a lehetőséget a nagy memóriakapacitással rendelkező, gyors gépek esetén használhatjuk ki igazán.

Különböző követelmények szerint kialakított adatállományok együttes kezelésére szemléletes példa lehet az irodalomfigyelés. Mégpedig olyan esetben, amikor nemcsak könyvekről, hanem újságokról is szó van. Ekkor külön adatbázist hozunk létre a könyvekhez, egy másikat az újságokhoz, majd ezt a kettőt összekapcsoljuk.

Az indexmezők alapján 25 különböző rendezettségben dolgozhatunk az egyes adatbázisokkal. Az aktuális indexmező egy adott értékére igen gyors lesz a keresés. Az adatbázisban történő lapozás során billentyűkombinációval aktivizálhatjuk a keresési folyamatot. Megadhatjuk a képernyős megjelenítés formátumát is: bizonyos mezőket érdemes kihagyni, így áttekinthetőbb és gyorsabb lesz az állomány böngészése. Természetesen a keresés és böngészés mellett mind képernyőn, mind nyomtatón megjeleníthetők a kívánt adatok, a választott sorrendben. A Zephyr ebben is sokat segít a dBASE-hez hasonló „report”-állományok alapján. Itt fejsorokat, lábsorokat, feliratokat adhatunk meg, és természetesen az adatmezők elrendezését is. Kényelmesen nyomtathatunk címkéket a Zephyrrel: itt is a megfelelő menüablakban adjuk meg a méretet, elrendezést, szöveget. A szöveg- és adatállomány tetszőlegesen keverhető.

A már említett szövegszerkesztővel körleveleket készíthetünk, amelyekbe a FoxPro-utasítások alkalmazásával az aktuális dátum és időpont is bekerülhet. A fenti nyomtatások bármelyikénél filterezhetjük adatállományunkat, azaz csak bizonyos jól definiált része kerül a listára, címkékre, illetve körlevélbe. A Zephyr outputja lehet képernyő, nyomtató vagy adatállomány. Érdekes lehetőség, hogy előre megadhatjuk bizonyos állományok képzésének feltételeit. Például, ha csak bizonyos napokon van szükségünk egyes listákra, azt egy állományban tartja nyilván a Zephyr. Minden indításkor betölti ezt az állományt, és ha az itt tárolt feltétel teljesül, jelzi a felhasználónak, és kinyomtatja a kérdéses listát.

Azokra is gondoltak a fejlesztők, akik szívesebben dolgoznak utasításmódban, nem szeretik az ablakos menüstruktúrát: ebben is nagyon hasonló a Zephyr kezelése a dBASE-éhez.

Összegezve: a Zephyr annyira jónak látszik, hogy a dBASE igazi vetélytársának is minősülhet. Nincs ugyan saját programozási nyelve, de a menü szerkezete és a rendelkezésre álló műveletek széles skálája ezt feleslegessé is teszi. Ha van Zephyrünk, nincs többé szükség külső szövegszerkesztőre sem.

Verebély Pálné

(A DOSshareware 1990/11-12. sz. alapján)

TANFOLYAM-AJÁNLATUNK

PC-DOS kezelői alaptanfolyam

Időtartama: 5 nap
Részvételi díj: 7500,– Ft

WordStar Professional 5.0 magyar

Időtartama: 5 nap
Részvételi díj: 8500,– Ft

NOVELL SFT felhasználóknak

Időtartama: 3 nap
Részvételi díj: 6600,– Ft

CLIPPER–WINDOW függvényrendszer

Időtartama: 5 nap
Részvételi díj: 8500,– Ft
Függvénykönyvtár: 12 900,– Ft
+ 25% 3225,– Ft

NOVELL SFT supervisoroknak

Időtartam
felhasználói rész 3 nap
supervisori rész 2 nap
Részvételi díj: 13 500,– Ft

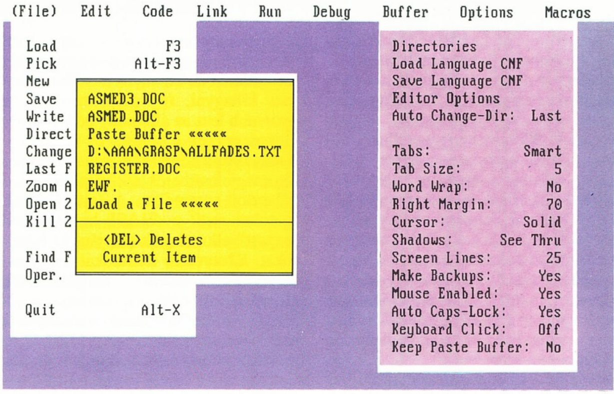
UNITRADE
Szervezési, kereskedelmi és Számítástechnikai K.F.T.

JELENTKEZÉS ÉS INFORMÁCIÓ AZ ÜZLETBEN

INFORMÁCIÓKÉRÉS: 13 ▼

ASMED, az ASseMbler EDitor
Kényelmesen programozni

Valószínűleg sok gépközeli nyelven programozó megirigyelte már a Borland Turbo programozási nyelveihez adott fejlesztői környezet kényelmét. (Nem kell kilépni a forráskód lefordításához, linkeléséhez, a hibás sorokra automatikusan pozicionál az editor stb.) Az assemblerben kódolók viszonylag szűkebb táboráról mintha mindenki megfeledkezett volna. Igaz, a Microsoft gondolt rájuk Quick C & Quick Assembler nevű csomagjával, de az bizony közel húszezer forint. Ezúttal egy egylemezes shareware programot ismertetünk, amelyet a SolarSoft Programkönyvtár nemrég megújult kínálatából szemeltünk ki.



Bár a program neve kissé arabusul hangzik: ASMED (mintha az első arab assembler program lenne), nyugodjunk meg, nem jobbról balra kell beírnunk a sorokat. A programot a Chicago Software nevű amerikai csapat bocsátotta közre, és mi már a legfrissebb változattal, a 3.0-val találkoztunk. Az ASMED integrált programfejlesztő környezet — egy editor ürügyén. Egérrel is vezérelhető, kezeli és kihasználja a PC EMS vagy akár extended memóriabővítését, egyszerre két ablakban is enged szerkeszteni. Egy 64 kilobájtos átmeneti (Cut/Paste) puffert kezel, melynek tartalmát blokkművelettel lehet létrehozni, és ezek után ugyanúgy editálhatjuk, nyomtathatjuk, mintha önálló fájl lenne. Az editorból közvetlenül hívhatunk compilert, linkert, debuggert, sőt kifejlesztett programunkat is futtathatjuk, mivel külső program meghívása előtt az ASMED 3,8 kB-ra „húzza össze magát".

Sok apró szolgáltatás áll a kényelmet szerető programozó (van másmilyen is?) rendelkezésére: lenyomott billentyűk scan-kódjának meghatározása, programozói kalkulátor, fájl- és szövegkeresés az összes meghajtót végigpásztázva stb.

A két ablak között természetesen átmásolhatjuk, illetve mozgathatjuk az előzetesen kijelölt blokk tartalmát. EGA vagy VGA kártyás gépen hangyányi méretű karakteres (43 vagy 50 soros) üzemmódba is átkapcsolhatunk. Alkönyvtárváltáskor szemléletes grafikus fastruktúra jeleníti meg az adott meghajtó katalóguságait. Használhatunk billentyűzetmakrókat is, a hosszabb leütéssorozatokat megjegyzi és tárolja az ASMED.

Ami a legjobban megragadott bennünket, az az automatikus Install/Setup program volt, amely átböngészte harddiszkjeinket, és minden egyes fordítót, linkert és debuggert felkutatott, sőt az ASMED-hez szükséges paraméterállományokat (*.CNF) is kitöltötte helyettünk. Az ASMED végül is minden valamirevaló compilerhez Borland-szerű fejlesztői felületet képes adni. Saját

KOGINFORM COMPUTER



AT-286/386/486 SZÁMÍTÓGÉPEK
MINDEN KONFIGURÁCIÓBAN MINDENKINEK!
KOGINFORM-COMPUTER Kft. 1042 Budapest, Tito u.10. Tel.: :1695146 Fax: 1695146, 1604209

INFORMÁCIÓKÉRÉS: 18 ▲

kedvenc programjaink közül is hetet megadhatunk, ezeket az ASMED menüből kínálja fel számunkra. Kedvünkre választhatunk árnyékot vető ablakokat (mert persze az ASMED teljesen menüvezérelt), azon belül is átlátszót vagy átlátszatlant, tilthatjuk vagy engedélyezhetjük az egér kezelését, a kurzor típusát, a képernyőn látható sorok számát, a színeket stb.

Billentyűparancsok az ASMED-ből

F1	— Show help
F2	— Save file
F3	— Load file
F5	— Zoom window(s)
F6	— Add/Switch window(s)
F7	— Block begin
F8	— Block end
F9	— Close window
F10	— Toggle main menu
Alt-F1	— Last help
Alt-F2	— Link code
Alt-F3	— Pick list
Alt-F4	— Run program (current)
Alt-F5	— Swap screens
Alt-F6	— Switch file (load previous)
Alt-F7	— Show next error
Alt-F8	— Show previous error
Alt-I	— Information on system/file — USEFUL
Alt-Q	— Show menu of errors
Alt-M	— Jump to main menu item()
Alt-V	— Show ASMED version information
Alt-X	— Leave ASMED
Alt-Z	— DOS shell
^J ^C	— Cut block to buffer
^J ^F	— Paste from buffer
^J ^P	— Print buffer
^J ^D	— Delete buffer
^G ^C	— Copy block to other window
^G ^M	— Move block to other window
^Q ^F	— Find
^Q ^A	— Find and Replace
^Q ^L	— Continue last find
Alt-1...0	— Execute macro (1...0)

Ragyogó helyzetérzékeny helpje van a programnak, amelyben többek között olyan speciális információk is fellelhetők, mint a processzor regiszterei, ASCII-tábla, BIOS-interruptok, DOS-függvények, a 8086-os processzor utasításai, egyes ismert fordítók paraméterezése (ASM, TCC, TP, TASM stb.). Maximum 15 elemet tartalmazó rámutatásos, menüs Pick-listát kínál fel a program Alt-F3-ra. Gyárilag támogatja az EXE2BIN konvertert. A fordítás (hiba) üzeneteit, figyelmeztetéseit az ASMED egy ún. Journal fájlba gyűjti. Kurzorunkat szép sorban a fordító által hibásnak nyilvánított sorokra teszi. A hibásnak ítélt sorok között az Alt-F7 és az ALt-F8 gombokkal mozoghatunk oda-vissza.

A program használatához szükséges állományok:
ASMED.EXE — a főprogram.
ASMED.OVR — az overlayállomány.
ASMED.HLP — a help szövege.
AR.EXE — a compilervezérlő kernel.
ASMED.VAR — az ASMED.EXE hozza létre.

A shareware program bevezető és záró képernyője figyelmeztet csupán arra, hogy ne feledkezzünk meg a regisztrációról, egyébként a program semmilyen korlátozást nem tartalmaz, és csak az editort tetszés szerint átszínező ASMCOLOR.EXE hiányzik.

-hj-

Az ASMED jellemző adatai

Maximális fájlméret: 64 kB (csonkítás nélkül)
Maximális vágó/betoldó puffer: 64 kB
Ablakok maximális száma: 2 + Journal
Egyszerre feldolgozható hibák: 40 + Journal
A Compiler Journal maximális mérete: 15 kB
Undo korlát: 1 sor
Makrók maximális száma: fájlonként 10, fájlok száma korlátlan
A Pick-List maximális mérete: 15
A Dos Shell memóriaigénye: 3,8 kB
Language CNF állományok száma: korlátlan
Könyvjelzők maximális száma: 10
Maximális blokkméret: 63 kB
Maximált sorhossz: 255 karakter

DataEase INTERNATIONAL

A **DataEase** adatbázis-kezelőt azoknak ajánljuk, akik értik az: angol, dán, finn, francia, holland, izlandi, magyar, német, norvég, olasz, orosz, portugál, spanyol, svéd nyelvek valamelyikét, ugyanis a **DataEase International** terméke ezeken a nyelveken is tud.

A **DataEase** egy egyedi vagy többfelhasználós (LAN) adatbázis-alkalmazást fejlesztő rendszer DOS környezetben, azoknak, akik a saját szakmájuk szakértői, akik színvonalas alkalmazásokat kívánnak egy-két nap alatt létrehozni, akik egyszerű nyilvántartásokat készítenek munkájuk segítéséhez, vagy akár azoknak, akik a számítástechnika professzionális alkalmazói.

Angliában 1990-ben a PC-s relációs adatbázis-kezelők közül a vásárlók több, mint 30 százaléka a **DataEase**-t választotta, jóval többen, mint akármelyik másik terméket.

A **DataEase International, Inc.** termékeinek magyarországi disztribútora a:

VT-SOFT Videoton Software Kft.
1033 Budapest, Vörösvári út 103-105.
Telefon: 180-3744
Telefax: 180-3750

VT-SOFT®
VIDEOTON SOFTWARE KFT

INFORMÁCIÓKÉRÉS: 21 ▼

Arcmaster fájlmester

A Sharc, a Shez és a Narc szoftverek után most újabb taggal bővült az adattömörítő programokkal rendelkező felhasználói felületek kínálata: megjelent az Arcmaster. Ezzel nemcsak megnézhetjük és egyenként kicsomagolhatjuk a tömörített állományokat, hanem oda-vissza konvertálhatunk a különböző archiválási formátumok között.

Az Arcmaster installálása roppant egyszerű. Az első elindításkor az „AM" begépelése és az F1 billentyű leütése után kiváló help-tartalomjegyzék jelenik meg, amelyből — a fel-le nyilakkal lépegetve — könnyen kiválaszthatjuk a minket érdeklő témát. A funkcióbillentyűket a mellékelt táblázatban foglaltuk össze.

A funkcióbillentyűket megismerve, az F2 leütésével megtudhatjuk, hogy az Arcmaster mely tömörítőprogramokat tudja kezelni (PKZIP/PKUNZIP, LHARC, PKPAK/PKUNPAK, ARC, ARCA/ARCE, PAK, LIST a 6.2 verziótól). Ezeknek a tömörítőprogramoknak a DOS keresési útvonalán kell lenniük ahhoz, hogy az Arcmaster a későbbiekben is megtalálhassa őket.

A fentiek ismeretében elkezdhetünk dolgozni. Próbáljunk először egy alkönyvtárat tömöríteni! Leütjük az F5 billentyűt, ekkor grafikusan megjelenik a könyvtári fastruktúra. A nyilakkal kiválasztjuk a kívánt könyvtárat, leütjük a Return billentyűt, ezzel behoztuk az Arcmaster menü ablakába ennek a könyvtárnak a tartalmát. Ezután a Ctrl-T billentyűkombinációval kijelöljük az összes tömörítendő állományt. Most kell kiválasztanunk a tömörítőprogramot. Az F2 funkcióbillentyűvel behozhatjuk a képernyőre a választékot. Válasszuk például az LHARC-ot. A tömörített állomány nevét úgy adjuk meg, hogy az F9 funkcióbillentyű leütése után begépeljük a megfelelő ablakba a kívánt nevet, legyen ez például TESZT. A kiterjesztés már automatikusan adott: „.LZH". Miután lezártuk ezt az ablakot, automatikusan megnyílik egy másik ablak, amelyik az LHARC paramétereit várja. Miután úgy döntünk, hogy az összes kijelölt állományt tömöríteni akarjuk, a folyamat a korábban megszokott módon megy végbe.

Mindeddig nem láttunk falrengető újdonságokat. De próbáljuk meg most az így tömörített TESZT.LZH állományt .ZIP kiterjesztésűvé átalakítani! Ez pofonegyszerű. A nyilakkal kiválasztjuk és a szóközbillentyűvel kijelöljük az .LZH-t. Alt-F9-re a .ZIP konvertáló formátumot választjuk, Alt-F10-zel pedig elindítjuk a konverziót. Ennek során az Arcmaster létrehoz egy átmeneti „$!$" könyvtárat, ebbe kicsomagolja az .LZH kiterjesztésű állományt, majd ZIP programmal újra tömöríti. Végül pedig szépen rendet csinál: kitörli az átmeneti könyvtárat.

Összefoglalva: nagyon jól használható segédeszköz az Arcmaster. Egyetlen megszorítás: főleg olyan felhasználóknak ajánlható, akik tudnak valamelyest angolul és van némi gyakorlatuk az adattömörítésben.

Verebély Pálné
(A DOSshareware 1991. februári száma nyomán)

F1	A help-tartalom megjelenítése	/	A parancs menü hívása
F2	Archiváló rendszer	INS	Az archív név kijelölése
F3	Meghajtóváltás	RET	Az állomány tartalmának megjelenítése
F4	Ablakváltás		
F5	A könyvtári fastruktúra grafikus megjelenítése	DEL	Állományok törlése
		SPACE	Egyenkénti kijelölés/kijelölés megszüntetése
F6	Állomány átnevezése		
F7	Az önkicsomagoló (SFX) állományok megmutatása	Ctrl-A	Automatikus felfrissítés (update)
F8	Az SFX állományok kiválasztása	Ctrl-D	Utolsó dátum
F9	Állománytömörítés	Ctrl-E	Editorhívás
F10	Állománykicsomagolás	Ctrl-M	Állománykiterjesztés szerinti kiválasztás
Ctrl-F1	Az archív állományok aktualizálása	Ctrl-L	Az LZEXE program behívása
		Ctrl-P	Nyomtatás
Ctrl-F2	Információ a tömörítőprogram keresési útvonaláról	Ctrl-R	Könyvtármegjelenítés (mint az F5)
Ctrl-F3	A legutóbb használt könyvtár megmutatása	Ctrl-S	Állománykeresés
		Ctrl-T	Az összes állomány kijelölése
Ctrl-F4	Víruskeresés a tömörített állományokban	Ctrl-U	Az összes állománykijelölés megszüntetése
Ctrl-F5	Könyvtárváltás	Ctrl-X	Képernyőfrissítés
Ctrl-F6	A kijelölt állományok aktualizálása	Ctrl-Z	A PKZIPFIX futtatása
Ctrl-F7	Reset bit	Alt-C	Állományok másolása
Ctrl-F9	Rekurzív „Zip"	Alt-F	A FormatMas behívása
		Alt-L	Az UNLZEXE behívása
Alt-F5	Az adott dátumhoz tartozó állományok kijelölése	Alt-M	Állományok mozgatása
		Alt-S	Egyes állományok archiválása
Alt-F9	A kívánt konvertálóprogram beállítása	Alt-T	A kijelölt állományok megjelenítése
Alt-F10	A konverzió elindítása	Alt-X	Kilépés a DOS-ba
F5 és F6	Könyvtár átnevezése	ESC	Kilépés a DOS-ba
TAB	Az aktuális ablak átváltása	Ctrl-Home/End	Újraindítás (startup)

Archív állományok konvertálása

A DOSshareware márciusi száma alapján lapunk mágneslemez mellékletén közreadott CONVERT.BAT batch program arra szolgál, hogy tetszőleges formátumú archív állományokat más archivált formátumúvá alakítsunk át.

A megfelelő archiváló és kicsomagoló programoknak a keresési út alkönyvtárában kell lenniük. Pillanatnyilag a következő archiváló és kicsomagoló programok által készített állományok kezelésére készült fel a CONVERT.BAT: ARC, ARCA, DWC, LHARC, PAK, PKARC, PKPAK, PKZIP 1.01 verziótól, ZOO, valamint az SFX önkicsomagoló archivált formára. SFX esetén az állomány nem SFX formátumú lesz, a legtöbb esetben ez nem is lenne kívánatos.

A régebbi archív állomány az átalakítás után ugyanolyan állománynévvel, de .OLD kiterjesztéssel található meg. Az SFX archív állományoknál a régi állománynév változatlan marad. A CONVERT.BAT hívásakor három paramétert kell megadnunk. A szintaktikus szabályokat könnyen felismerhetjük, ha csak egyszerűen meghívjuk a CONVERT-et úgy, hogy semmi feladata ne legyen.

A paramétereket az alábbi sorrendben kell a DOS prompt után megadnunk:

1. Annak az archív állománynak a neve, amelyet át akarunk alakítani, de ez ne tartalmazza a kiterjesztést.
2. Az a program, amellyel ez az archív állomány eredetileg készült, tehát például ARC, PKPAK stb. Az SFX archív állományoknál csak SFX-et kell írni.
3. Az a program, amelynek a formátumára át akarjuk alakítani az archív állományt, tehát például PKZIP, LHARC stb.

A szerző nem garantálja, hogy a program minden körülmények között tökéletesen működik. Csupán azt állítja, hogy nála semmilyen problémát nem okozott. MS-DOS 3.3 alá íródott.

A programmal kapcsolatos észrevételeiket az alábbi címen közölhetik a szerzővel:

Joachim Meyn
c/o Telefonmarketing
Angelika Meyn
Postfach 1305
D-6909 Walldorf

Új helyen a Cédrus!

A Cédrus Informatikai Részvénytársaság irodáinak, kiadói és szerkesztőségi részlegeinek új címe:

1251 Budapest XI., Karolina út 17.

Telefon: 166-2111
Fax: 185-2221

Az új Shez

A krónikus pénz- és winchesterhiányban a tömörítőprogramok már-már olyan tartozékaivá válnak a számítástechnikának, mint a Norton Commander. Néhány hete jelent meg az USA-val online kapcsolatban lévő BBS-eken a Shez 6.1 új verziója, annak is a teljes kiadása.

A programot — meglepő módon — már regisztrálva is terjesztik Európában. Mindkét változatot alaposan tesztelve, a következő alapvető dolgok derültek ki. A szabad verzió jóval gyorsabb, mint regisztrált testvére. Ugyanakkor nincs benne a kezelhető állományok számára vonatkozó megkötés sem. Nyugodtan átinstallálható, de az installálás során a sorszámot beállító menübe nem szabad belépnünk.

Az új, 6.1-es változaton érződik, hogy írói gyökeres változtatásra készülnek. Csak nem térnek át a szabadszoftverekről a professzionális, kereskedelmi rendszerekre?

A szoftvert felkészítették a nálunk is terjedőben lévő ARJ tömörítőprogram kezelésére is. Lényeges változás a korábbi verzióhoz képest, hogy az LHARC program jelenlegi formájában megszűnt, a programot a Corel Draw gyártója professzionális felhasználásra megvette, és a japán szerző az utolsó LHARC-verzióval teljesen kompatibilis LHA programmal örvendeztette meg a számítástechnikai rendszerek használóit. (Lásd erről részletesen a hónap témájának anyagát lapunk 2—14. oldalán.)

A korábban megszokott szolgáltatások, mint például a McAfee vírusszkenner kezelése, a futtatás tömörített állományból stb. változatlanul megmaradtak. A tapasztalat szerint a program Novell hálózatokon is kifogástalanul működik.

kis

Jön, jön, jön... És magyar!

A SolarSoft magyar szekciója ismét bővült néhány igen értékes programmal. A DIRI 5.01-es változata csupán felújítás, de az angol-magyar, magyar-angol szószedet mintegy 9500, a német-magyar, magyar-német pedig több mint 7200 szópárt tartalmaz.

A VÁM 91 nemcsak látványos, hanem hasznos program is. Akiknek gyakran kell megmérkőzniük a Vám- és Pénzügyőrség bürokráciájával, azok tudják, mit jelent egy vámstatisztikai űrlap pontos kitöltése. Ezt könnyíti meg a VÁM 91.

A GIB_DEMO egy teljes magyar ékezetes angol-magyar, magyar-angol szótár demonstrációs változata. A fonetikai jelek megjelenítése érdekében kizárólag grafikus üzemmódban használható, így csak CGA, Hercules, EGA és VGA kártyás gépeken indul el, és a klasszikus MDA monokróm kártyával nem kompatibilis. A program rezidens üzemmódban is indítható, ekkor bármilyen más program futása közben is előhívható az általunk megadott billentyűkombinációval. A demó – terjedelmi okok miatt – csak az A betűvel kezdődő szavakat, szócikkeket tartalmazza. Hogy mégis miért érdemes megvásárolni a demólemezt? A szerzőkkel folytatott tárgyalások alapján, aki hajlandó pénzt áldozni a shareware lemezre, az visszakapja annak árát a kereskedelmi változat megvásárlásakor (4000 Ft + áfa). Tisztességes ajánlat, nincs kockázat, a GIB egy és két lépcsőben történő vásárláskor is ugyanannyiba kerül.

A PASSIV egy német nyelvtant oktató és sulykoló tesztprogram. Addig ismételgeti a nyelvtani szabályokat, Amíg azokat a tanuló százszázalékosan el nem sajátítja. A lemez a passzív igeragozást gyakoroltatja.

A játékok kedvelőinek jó hír az EGA kártyát igénylő Rabló Römi nevű program.

A Turbo DBU (TDBU) program a Clipperhez mellékelt, közismert DBU jelentősen továbbfejlesztett változata, amelynek révén Clipper adatbázisokkal és indexállományokkal tudunk kényelmesen dolgozni (*.DBF, *.NTX). Néhány egyedi szolgáltatása: visszacsatolt relációk, felhasználóbarát böngésző (browse), indexkulcsok gyors cseréje, nyomtatási modul automatikus fejléc-generátorral, 5 megadható összegfokozattal, a felhasználó által megadható műveleti mezőkkel (mintha Lotus 1-2-3-ban vagy Quattróban lennénk).

Aktuális napi menedzseri feladatokat vállal a The Great Speculator nevű programcsomag. A kulcsszó: menedzser. Menedzsernaptár, partneradatbázis, titkosítás. Részvények, kötvények árfolyamainak, egyéb mutatóinak kezelése, megjelenítése, összehasonlítása szemléletes grafikonok segítségével, deviza- és keresztárfolyamok, árutőzsdei adatok nyilvántartása, röviden: komplex bróker-szoftver. A programból csak a részletes help hiányzik.

A FLOPPY.LAP júliusi számából

Objektumorientált könyvtár Turbo Pascal 5.5-höz

●

Színezzünk magasszintű nyelvekből!

●

Fotók DIGICAM-mel

●

TEGL

●

GYÓGY(H)ÍR rovat

SolarSoft sikerlista

Az 1991. áprilisi és májusi eladások alapján

	No.	Programnév	Db	Programleírás
1.	421	PKZ110 & PKLITE & SHEZ	1	A „sűrítés" Norton Commandere
2.	319	SCAN74 & OTHERS	1	McAfee-féle vírusmegelőző, -detektor és -ölő
3.	470	MULTI-EDIT 5.0	1	A világon a legjobbnak tartott editor
4.	494	TEGLP WIND. TOOLKIT	1	Ikongrafikus felület, ikoneditor TP-hez
5.	327	LHA 2.10 & LHICE	1	Japán szupertömörítő/önkicsomagoló program
6.	432	LZEXE & LIST 7.5e	1	EXE kompresszor, Vernon Buerg LIST PLUS-a
7.	385	QEDIT ADV. 2.1, QHELP	1	A legkisebb, de nagytudású editor
8.	475	NEWSPACE	1	Merevlemezünk kapacitását megduplázza!
9.	096	AS-EASY-AS 4.00p	1	Lotus-kompatibilis egyszerű táblázatkezelő
10.	463	GAMES FOR MS WIND.	1	10+1 játék MS Windows 3.0 alá
11.	477	BACK & FORTH	1	Memóriamenedszer: 20 program egyszerre!
12.	383	4DOS V3.01a	1	COMMAND.COM pótló DOS héj: 50 új parancs
13.	466	SKYGLOBE STAR GAZER	1	Mozgó, színes csillagtérkép!
14.	474	JORJ POP-UP DICT.	2	58 000 szavas angol értelmező szótár
15.	480	GRASP 1.10C	1	Látványos animáció- és demókészítő program
16.	425	POP-DBF 1.1 & DLITE	1	Tárrezidens dBASE (EDIT/BROWSE/DISP stb.)
17.	468	SUPER ASSEMBLER ED.	1	TASM-ra kihegyezett programeditor
18.	485	BASIC COMPILER	1	Két ragyogó fordító — editorral
19.	488	COMPILER TUTORIAL	1	Készítsünk compilert! (*.C)
20.	304	TURBO TECHNO JOCKS	2	Szuper Turbo Pascal unitok forrásaikkal!
21.	435	OPTIKS 2.18 & ICONVERT	1	PCX,PIC,GIF,TIF,GEM,MAC... grafikus konverterek
22.	502	C++ TOOLKITS #5	1	Szenzációs objektumorientált kiegészítések
23.	495	TEGLC WIND. TOOLKIT	1	Ikongrafikus felület, ikoneditor TC-hez
24.	484	SR-INFO	2	dBASE-kompatibilis fejlesztőrendszer
25.	490	SURPASS	1	Önálló Pascal fordító, editorral
26.	472	SHARESPELL	1	Bővíthető, önálló helyesírás-korrektor
27.	478	XTREE 2.0E	1	Az ismert, kisméretű, gyors fájlmenedszer
28.	497	DESMET C COMPILER	1	Teljes C fordító, linker + assembler
29.	492	OOP #2	1	Objektumorientált prog. TP 5.5
30.	476	LOGIT!	1	A géphasználatot naplózó TSR

INFORMÁCIÓKÉRÉS: 28 ▲

SZÁMÍTÁS-, IRODA-, INFORMÁCIÓTECHNIKAI CÉGEK, FIGYELEM!

EGY SZAKKIADVÁNY, AMELY 10 000 FELHASZNÁLÓHOZ JUT EL!
RENDSZER, TARTALOM, FORMA, SZÉLES KÖR = HATÉKONY INFORMÁCIÓÁRAMLÁS!
EGY KATALÓGUS, AMELYBEN BENNE KELL LENNI!

Szerepeljen az első igazi számítás-, iroda-, információtechnikai szakkiadványban, amely 10 000 FELHASZNÁLÓHOZ JUT EL, EBBŐL 5000-HEZ DÍJTALANUL!

Kiadványunkat rendszeresen, évente kétszer jelentetjük meg, melyből 5000 példányt postán küldünk szét a termelő, a kereskedelmi és a szolgáltató cégek vezetőinek, szakembereinek. További 5000 példányt pedig a COMPFAIR kiállításon — 1991-ben — 200 forintért árusítunk, hogy egy szélesebb szakmai kör is hozzájuthasson.

A KIADVÁNNYAL ÁTFOGJUK A SZÁMÍTÁS-, AZ IRODA-, ÉS AZ INFORMÁCIÓTECHNIKA EGÉSZ TERÜLETÉT, ANNAK ÉRDEKÉBEN, HOGY A RÉGI ÉS A LEENDŐ FELHASZNÁLÓKNAK AZ ÚJDONSÁGOKAT ÉS A TELJES VÁLASZTÉKOT BEMUTASSUK!

A jobb áttekinthetőség érdekében a kiadványt egy részletes TEMATIKUS TÁRGYMUTATÓVAL is ellátjuk.
A kiadvány fejezeteinek és egyben a tematikus tárgymutatónak a címei:

I. SZÁMÍTÁSTECHNIKA

1. Hardver
   - Számítástechnikai hardver
   - Folyamatirányítás
   - Szerviz
   - Oktatás
   - Szervezés

2. Szoftver
   - Magyar gyártmányú szoftver
   - Külföldi szoftver
   - Kisvállalkozók által készített szoftverek

II. IRODATECHNIKA — TELEKOMMUNIKÁCIÓ

III. EGYÉB, KAPCSOLÓDÓ TERÜLETEK:
   - Például vonalkódtechnika

ÁRAINKKAL A RÉSZLETESEBB, TÖBB INFORMÁCIÓT ADÓ CÉGEKET TÁMOGATJUK:

1/2 B/5	oldal	összesen	25 000 Ft + Áfa	50 000 Ft/oldal
1 db B/5	oldal	összesen	40 000 Ft + Áfa	40 000 Ft/oldal
2 db B/5	oldal	összesen	70 000 Ft + Áfa	35 000 Ft/oldal
3 db B/5	oldal	öszesen	90 000 Ft + Áfa	30 000 Ft/oldal
4—5 db B/5	oldal	összesen	140 000 Ft + Áfa	28 000 Ft/oldal
6—7 db B/5	oldal	összesen	175 000 Ft + Áfa	25 000 Ft/oldal
8—10 db B/5	oldal	összesen	200 000 Ft + Áfa	20 000 Ft/oldal

Első belső, hátsó belső borító: 70 000 Ft + Áfa.

Árainkkal külön is TÁMOGATJUK A GMK-KAT, BT-KET, KKT-KAT ÉS AZ EGYÉNI VÁLLALKOZÓKAT: ebben az esetben, 400 karakter terjedelemig 2000 Ft + Áfa árért is bekerülhetnek a kiadványba.

EGYÉB FELTÉTELEK:
— A részvénytársaságoknak, a vállalatoknak, a kft.-knek, a kisszövetkezeteknek a szerződés megkötésekor át kell utalniuk a szerződött összeg 50 százalékát, a további 50 százalékot pedig a kiadvány átvétele után.
— A kisvállalkozók csak a teljes összeg átutalása esetén szerepelhetnek a kiadványban.

A KÉZIRAT LEADÁSÁNAK HATÁRIDEJE: 1991. AUGUSZTUS 1.
A KIADVÁNY MEGJELENÉSE: COMPFAIR '91 — 1991. OKTÓBER 10.

MEGRENDELÉSÜKET AZ ALÁBBI CÍMEN, ILLETVE TELEFAXSZÁMON VÁRJUK:

HÍRVIVŐ BT.

1081 BUDAPEST, BEZERÉDI U. 6. I. 6.
Telefax/üzenetrögzítő: 178-4421

INFORMÁCIÓKÉRÉS: 19 ▶

SZERZŐDÉS

Megrendelünk a Hírvivő BT. és a MGÉSZV kiadásában megjelenő INFORMÁCIÓTECHNIKA '91 katalógusában:

☐	1/2 B/5	oldalt	összesen	25 000 Ft + Áfa árban
☐	1 db B/5	oldalt	összesen	40 000 Ft + Áfa árban
☐	2 db B/5	oldalt	összesen	70 000 Ft + Áfa árban
☐	3 db B/5	oldalt	összesen	90 000 Ft + Áfa árban
☐	4—5 db B/5	oldalt	összesen	140 000 Ft + Áfa árban
☐	6—7 db B/5	oldalt	összesen	175 000 Ft + Áfa árban
☐	8—10 db B/5	oldalt	összesen	200 000 Ft + Áfa árban

A minden megrendelőnek járó 1 db ingyenes példányon felül megrendelek ...... db kiadványt (Ára 200 Ft).

ÖSSZESÍTVE:

KATALÓGUSOLDAL ÖSSZESEN: ....Ft

ELSŐ BELSŐ BORÍTÓ: ....Ft

HÁTSÓ BELSŐ BORÍTÓ: ....Ft

+ 25% Áfa: ....Ft

KIADVÁNY (0% Áfa): ....Ft

MINDÖSSZESEN: ....Ft

A megrendeléssel egyidőben gondoskodom a hirdetési összeg 50 százalékának a MEZŐBANK RT. 219-98911-0112-700 számú bankszámlájára történő átutalásról — INFORMÁCIÓTECHNIKA '91 megjelöléssel —, részteljesítésként. A fennmaradó 50%-ot a kiadvány kézhezvétele után 10 munkanapon belül átutalom a fenti számlára.

Dátum: ....

....
cégszerű aláírás
bélyegző

Pontos cím: ....

Ügyintéző neve: .... Telefonszám: ....

Bankszámlaszám: ....

Egyéb megállapodás: ....

SZERZŐDÉS

Megrendelünk a Hírvivő BT. és az MGÉSZV kiadásában megjelenő INFORMÁCIÓTECHNIKA '91 katalógusában max. 400 karakter hirdetési szöveget és ...... darab kiadványt.
A megrendeléssel egyidőben a 2000 Ft hirdetési díjat és a ...... darab kiadvány árát a MEZŐBANK RT. 219-98911-0112-700 számú bankszámlájára rózsaszín postautalványon befizettem.

Dátum: ....

....
cégszerű aláírás
bélyegző

Pontos cím: ....

MEGRENDELÉS

Megrendelünk a Hírvivő BT. és az MGÉSZV kiadásában megjelenő INFORMÁCIÓTECHNIKA '91 katalógusból ...... példányt, utánvéttel.

Dátum: ....

....
cégszerű aláírás
bélyegző

Pontos cím: ....

INFORMÁCIÓKÉRÉS: 20 ▲

NETREND RT

1089 Budapest, Elnök u. 1.
Tel: 113-8217; 133-4760 • Fax: 113-9537

XT-10 számítógép	
– 640 kilobájt RAM	
– Multi I/O kártya	
– 360 kbájtos FDD	
– 101 gombos billentyűzet	29 800
– 83 gombos billentyűzet	3 500
– 101 gombos billentyűzet	3 800
AT 286-12/16 számítógép	
– 1 megabájt RAM	
– FDD/HDD vezérlő + S/P kimenet	
– 1,2 Mbájtos FDD	
– 101 gombos billentyűzet	35 900
AT 286-16/21 számítógép	**38 000**
NEAT 286-16/21 számítógép	**39 900**
NEAT 286-20/25 számítógép	**51 900**
NEAT 286-24/32 számítógép	**53 900**
AT-386-20/25 MHz,	
– 2 megabájt RAM	
– FDD/HDD vezérlő + S/P kimenet	
– 1,2 Mbájtos FDD	
– 101 gombos billentyűzet	79 200
AT 386-25/33	**85 100**
AT 386-25/43, 64 kB cache	**103 400**
AT 386-33/58, 64 kB cache	**103 300**
486-25/117, 4 MB RAM	
– 128 kB cache	248 000
486-33/147, 4 MB RAM	
– 128 kB cache	290 050
Monitor-csatolókártyák:	
Monokróm	1 700
Color	2 400
EGA 800x600	5 500
VGA 800x600	9 600
VGA 1024x768	11 800
VGA 1024x768	15 600
VGA 1024x768	16 500
Monitorok:	
Egyszínű (borostyánsárga)	10 400
Egyszínű (papírfehér)	10 900
EGA	28 900—31 900
VGA (1024x768)	35 500
VGA egyszínű, 1024x768	22 000
VGA Multisync	46 900
MFM, ESDI és IDE winchesterek nagy választékban!	
Hálózati terminál:	
NEAT 286-12	48 750
NEAT 286-16	56 500

Szünetmentes áramforrások:	
UPS 400 VA NOVELL	39 900
UPS 550 VA NOVELL	32 000
UPS 600 VA NOVELL	45 800
UPS 1 kVA	54 600
UPS 1,2 kVA	98 500
UPS mon. kártya	7 500
Nyomtatók:	
FX-850	49 500
FX-1050	48 750
LQ-850	76 400
LQ-2500+	129 000
DFX-500	183 500
DL 5600 (színes)	195 000
HP LASERJET III.	199 00
Memóriabővítő kártyák:	
286/2 megabájt	9 900
286/3,5 megabájt	10 800
386/2/8 megabájt	15 000
RAM-ok:	
4164-10	140
41464-08	290
41256-08	160
41256-06	280
44256-08	780
511000-10	750
511000-08	760
Koprocesszorok:	
80287-10	14 000
80287-20	44 000
80387-20	44 000
80387-35	53 800
80387-33	69 900
Modemek:	
2400 baud belső	10 900
2400 baud külső	14 500
2400 baud MNP-5	18 500
Telefax/modem:	
9600/2400 baud	32 500
8 felhasználós hálózati telefax	75 000
Egerek, scannerek:	
GM-6000 egér	4 950
Microsoft egér	16 000
Logitech soros	5 440
Logitech scanner	21 700
Catchword karakterfelismerő program	22 900
HP SCANJET PLUS	259 000

Plotterek:	
SEKONIC 450	115 600
HP 7475A A3	179 000
MUTOH 910E	1 380 000
ARCnet kártyák:	
8 bit LIN DATA	4 800
8 bit ZOT	5 400
8 bit SMC	9 900
16 bit LIN DATA	9 800
16 bit ZOT	10 500
ETHERNET kártyák és tartozékok:	
8 bit NE-1000	9 900
8 bit DE-100	14 500
8 bit DE-150	19 600
16 bit NE-2000	12 900
16 bit DE-200	16 500
HUB-ok:	
Passzív HUB E4	2 000
Aktív HUB (int 4)	7 800
Aktív HUB (ext)	14 000
Aktív HUB + kártya	12 000
Csatlakozókábel	1 500
BOOT-EPROM	2 000
Ethernet-kiegészítők:	
Transceiver	35 000
Transceiver BNC	25 000
Repeater (2 port)	92 000
Repeater (4 port)	148 000
Hálózati szoftverek:	
Novell NetWare	
V.2.2/5	72 000
V.2.2/10	161 000
V.2.2/100	444 000
V.3.11/20	282 000
V.3.11/100	565 000
V.3.11/250	1 010 000
NACS	110 000
Asyncron Remote Bridge Program	29 600
D-Link Lansmart op. rendszer	34 500
D-Link Bridge	29 600
ACS D-Linkhez	28 000
Remote Access	19 600
Screen monitor	15 000
LAPTOP-ok:	
LT-3400 (NEAT)	
40 MB HDD	199 000
CP-8100V (386)	
100 MBB HDD	299 000

És még sok minden egyéb...
Keresse termékeinket Székesfehérváron, az IZISZ Kft.-nél is!
Székesfehérvár, Palotai út 139. Telefon: (22)16-049.
A Netrend Rt. a Novell Inc. hivatalos dealere.
Vállalkozunk komplett hálózati rendszerek szállítására, igény szerinti kiépítésben.
Komplex rendszerfelügyelet (hálózati is), szaktanácsadás, hardver- és szoftverkarbantartás.
CAD, DTP rendszerek kiépítése, szükség esetén üzemeltetése. Kérje részletes tájékoztatónkat!
Áraink az ÁFA-t nem, de a 6 hónap csereszavatosságot tartalmazzák, egy év csereszavatosság — plusz öt százalék. Készpénzfizetés esetén öt százalék kedvezmény!
Önkormányzatok, oktatási intézmények, egészségügyi szervezetek részére 5 százalék kedvezmény!
Kedvező lízingfeltételek!

INFORMÁCIÓKÉRÉS: 15 ▼

Bábeli nyelvzavar

Adatra vagy objektumra orientálva

A gyakorlati tapasztalatok alapján bátran mondhatjuk, hogy a CA- (computer aided) rendszerek használhatóságát jelentős mértékben befolyásolja az adatleírás, az adattárolás és a visszakeresés választott módszere. Tartalmas tervezési adatbázisok alkalmazásával és a számítógéppel támogatott rendszerek közötti adatkapcsolatok révén számottevően növelhető a gyártmány- és gyártástervezés eredményessége.

A számítógéppel segített tervezésben alapvetően három adatbázisfajtát lehet megkülönböztetni. A legáltalánosabb az archív adatbázis, amely mérnöki dokumentációk tartós tárolására szolgál. Szerepelhetnek benne például rendelési specifikációk, ajánlati tervek, korábbi termékrajzok, saját gyártmányleírások, költségszámítási jegyzékek és így tovább. A tervezési tevékenységhez ennél szorosabban kötődnek a mérnöki adatbázisok, amelyek szabványelőírásokat, anyagféleség-nyilvántartást, kereskedelmi termékleírásokat, tételjegyzékeket, normaadatokat, költségadatokat, típusterveket stb. tartalmaznak.

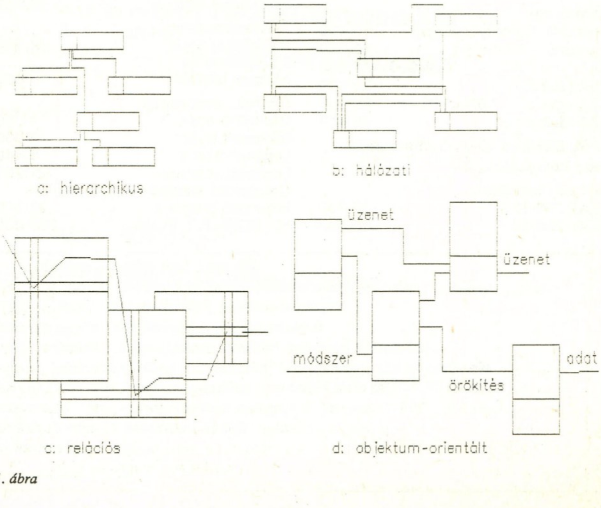
A harmadik adatbázisféleséget a projekt adatbázisok képviselik, amelyek három további típusba sorolhatók. Ezek a tárolt adatok jellemző szemantikai tartalma alapján különböztethetők meg. A rajzi adatbázis a tervezett objektumot 2D-s rajzi állomány formájában írja le, ennélfogva a tárolt adatok gyártási célra közvetlenül (azaz utófeldolgozás nélkül) nem használhatók fel. A modell adatbázisok háromdimenziós huzalvázak, felület- vagy testmodellek morfológiai adatait tartalmazzák. Ezek az adatok elvileg közvetlenül felhasználhatók elemzési, technológiai előkészítési, szimulációs vagy gyártási célokra. A harmadik adatbázistípus a termékadatbázis, amely a terméket és a megvalósítási folyamatot jellemző adatok rendszerezett nyilvántartására és archiválására egyaránt kiterjed.

Elszakadva az alapoktól

A fentebb említett projekt adatbázist a szokványos ügyviteli-nyilvántartási adatbázisoktól elsődlegesen az különbözteti meg, hogy: (a) sokféle adategységet tartalmaz; (b) gyakran van szükség az adatok módosítására; (c) a megváltoztatott adatok visszahatnak a meglévő tárolási struktúrára; (d) az adatokat nem csak közvetlen hivatkozás alapján kell elérni; (e) dinamikus tárolásra és visszakeresésre van szükség; (f) egyazon adathoz többféle értelmezést is hozzá kell rendelni; (g) az adatokat nagy pontossággal kell ábrázolni; továbbá (h) az adatbázis redundanciamentességét biztosítani kell.

E követelményeknek egyidejűleg általában csak kompromisszumok árán lehet megfelelni a jelenleg ismert adatkezelési technikákkal. Ugyanakkor erőteljes kutatások folynak az úgynevezett intelligens adatbázisok fejlesztésére, amelyek az adatokat nemcsak szintaktikusan kezelik, hanem bizonyos mértékben jelentéstartalmukat és kapcsolataikat is megértik.

A CAD adatkezelés lényegének megértéséhez célszerű áttekinteni azokat a fogalmakat és megoldásokat, amelyek napjainkban CAD-környezetben felmerülnek, illetve alkalmazásra kerülnek. Látni kell mindenekelőtt, hogy a számítógéppel segített tervezésben az adatmodellezés négyszintű leképezéssel valósul meg. Először is a felhasználó (tervező) a valós vagy elképzelt világra kialakít valamilyen szemléleti módot. Ezt fizikai vagy koncepcionális sémának nevezzük. A tervezőnek ebből kiindulva, a lényeges sajátosságokra vonatkozó adatok megragadásával absztrakt (logikai) modellt kell kialakítania, amit adatstruktúra sémának nevezünk. A CAD-rendszer adatkezelője az adatállományokat lényegében az adatstruktúra sémája alapján rendeli hozzá a fizikai fájlokhoz. Ezt adattárolási sémának nevezzük. A negyedik séma az adatoknak a számítógépes hardvereszközök fizikai szintjén megvalósuló kezeléséhez kapcsolódik, és tárolóhely-ki-



1. ábra

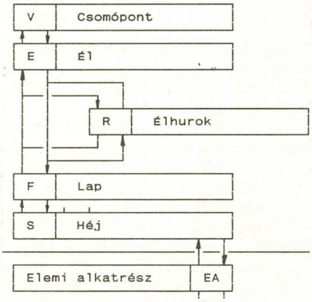
osztási sémaként értelmezhető. E négylépcsős leképezés, ami végső soron a jelenleg használt Neumann-elvű számítógépek információfeldolgozási mechanizmusából adódik, valamilyen formában minden CAD-rendszerben megvalósul.

Az adatszervezés legelemibb formája a soros lista szervezése. Ez nem képes az adatok között (a sorrendiségen túlmutató) kapcsolatok kezelésére. A hatékony tárolás és visszakeresés viszont az adategyüttesek struktúrába szervezését igényli, amelynek alapvetően négy formája terjedt el. Ezek az elfogadottá válásuk sorrendjében a hierarchikus, a hálózati, a relációs és az objektumorientált adatkezelés (1. ábra). Az említettek a CAD-ben való alkalmazás szempontjából bizonyos sajátosságokkal rendelkeznek, amelyeket érdemes megismerni. A négy adatkezelési módszer különbözősége elsődlegesen nem a kezelt adategységek különbözőségére vezethető vissza, hanem az adategységek között lévő kapcsolatok megragadásának módjára.

A hierarchikus adatstruktúrában az összetartozó adategységek közötti kapcsolat 1:N típusú. Ez azt jelenti, hogy egy, a hierarchia magasabb szintjén elhelyezkedő adategységhez (a szülőhöz) N darab különböző alacsonyabb szintű adategység (gyerek) tartozhat. Az adategységek ilyenfajta kapcsolódása fastruktúraként szemléltethető, amely meglehetősen rugalmatlan, mivel az adategységek közötti kapcsolatok leírása beépül az adatstruktúrába. E kapcsolatok újradefiniálása a séma megváltozását eredményezi. A hierachikus tárolási formában az adategységek rövid idő alatt, többletráfordítások nélkül elérhetők, ami bizonyos mértékben ellentételezi rugalmatlanságát.

Mivel a gépészeti objektumok geometriai modelljeinek adategyüttese közvetlenül általában nem rendezhető hierarchikus struktúrába, e megközelítés — a CSG-fák leírásának kivételével — nem nyert széles körű alkalmazást a geometriai modellezésben. Ugyanis, ha többszörös adategység-kapcsolatokat kívánunk hierarchikus struktúraként leképezni, akkor redundáns adategységtárolást kellene megvalósítani. Ez viszont egyrészt következetlenség forrása lehet, másrészt nagyobb tárolókapacitást igényel. A hierarchikus struktúra mindamellett kitűnő tárolási struktúra lehet azokban az esetekben, amikor az adategyüttes hierarchia relációkkal lefedhető. A gépészeti gyártmányok alkatrészei közötti kapcsolatok például ilyen jellegűek. Az archív és mérnöki adatbázisok szervezésénél emiatt gyakori tárolási séma. Geometriai valóság szemléltetésére viszont előnyösebb a hálózati struktúra.

A hálózati adatstruktúrában az összetartozó adatok között a kapcsolat M:N jellegű, vagyis bármely adategység bármely adategységgel kapcsolódhat. A hálózat a hierarchiánál bonyolultabb, de mégis rugalmasabb és hatékonyabb tárolóhely-kihasználást tesz lehetővé. Az adategységek közötti kapcsolatok ebben az esetben is beépülnek a tárolási struktúrába. A geometriai modellezésben a relációk logikai tartalmát a csomópontok, élek, élhurkok, lapok



2. ábra

és héjak közötti megfeleltetések adják (2. ábra). A hálózati jellegből adódóan hosszabb adatelérési idővel kell számolni. A geometriai modellezésen kívül a hálózati adatkezelést előszeretettel alkalmazzák a kereskedelmi forgalmazású mérnöki adatbázis-fejlesztő környezetek is.

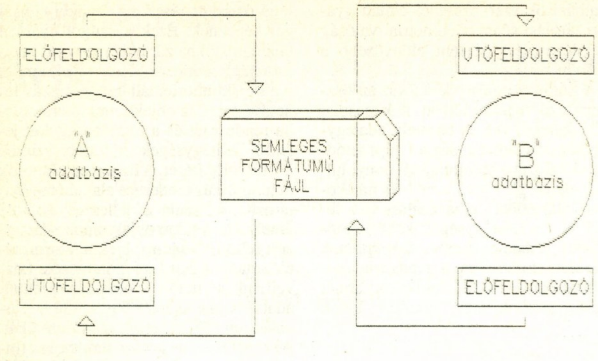
Az eddig említett adatkezelési megoldásokkal összevetve az ún. relációs séma újabb előnyös lehetőséget kínál. E koncepció alkalmazásakor az adategységeket kapcsolattáblákban helyezik el, ami előszervezést biztosít. Az adatok közötti kapcsolatot az adatbáziskezelő vagy a felhasználói CAD szoftver definiálja. Így az adategységek és a táblázatok különböző külső sémák szerint hozhatók kapcsolatba, ami nagyfokú rugalmasságot teremt. Az adathozzáférési sebesség viszont kedvezőtlen, bár különböző kifinomult technikákkal javítható.

Az objektumorientált adatkezelés sztereotipizálható dolgokra vonatkozó adategyüttesek szemantikus kezelésére szolgál. Míg az adatorientált szemléltetések a lényeges jellemzőket tényadatokként absztrahálva írják le a dolgokat, az objektumorientált szemléltetés esetében az adategységek és a közöttük lévő összefüggések egy logikai egységet képeznek. Ezek az angol nyelven találóan „tokba zártnak" nevezett adat- és relációszerkezetek az objektumok. Az objektumorientált programozás lehetővé teszi az objektumok hierarchiába rendezését és a tulajdonságokat jelentő adategységek hierarchiaszintek közötti örökítését. A fizikai objektumok taxonomikus rendezése alapján meghatározhatók azok a jellemző adatok, amelyek a logikai objektumok adatmezőit feltöltik. Bár az objektum információtartalmát hordozó adatmezők közvetlenül elérhetők, az objektumorientált adatsémában kezelésükre külön eszközöket rendszeresítettek: a módszereket. Az objektumhoz tartozó módszerek (tulajdonképpen adatrelációk és függvények) lehetővé teszik az egyes adatmezők értékeinek származtatását, de ugyanez külső módszerek számára nem megengedett. A módszerek segítségével az objektumok valamilyen „viselkedést" mutatnak, ha megfelelő üzenetet kapnak. Az objektumorientált szemlélet nem új keletű, hiszen közel 25 éve ismert. Korábban főleg szimulációs feladatokra alkalmazták, de napjainkban tért hódít a grafikus rendszerek és a mérnöki adatbázisok fejlesztésében is.

Kifelé az adatokkal...

Hozzátartozik az igazsághoz, hogy a CAD-rendszerek fejlesztői gyakran teljesen eltérnek a korábban tárgyalt tipizált megoldásoktól, amelyek gyökerei az általános adatfeldolgozásba nyúlnak vissza. Például a mikroszámítógépes CAD-rendszerek többsége projekt adatbázisként egyetlen fájlt kezel. Ez lehet akár rekordszerkezetű (bináris) szövegfájl, de általában alternatív alapegységrekordokból felépülő adatfájl. A fájlban geometriai és annotációs alapegységeket képeznek le olyan sorrendben, ahogy azokat a tervező létrehozta. A rendszerek egyik csoportja a tervezés folyamán a memóriában dolgozik, és a mágneslemezes háttértárolón csak a munka befejezése után archiválja a rajzi adatállományt. A másik csoport — részben a memória felszabadítása, részben az időszakos mentés céljából — általában automatikusan felülírja a háttértárolón lévő fájlokat. A geometriai modellező rendszerek többsége a korábban említett hálózati vagy objektumorientált adatkezelési sémák alkalmas módozatain alapuló adatbázis-kezelést valósít meg.

A fejlesztők sajnálatos módon nem hozzák nyilvánosságra, hanem csak kebelbarátaikkal osztják meg az adatbázis



3. ábra

struktúrájára és a kezelt adatok leírására vonatkozó információkat. Ennek eredménye természetesen egy sajátos bábeli nyelvzavar. A CAD-rendszerek nem képesek egymás adatfájljának közvetlen fogadására, és jó, ha ugyanazon rendszer későbbi változatai képesek az előzőek fájljait kezelni.

Géprajz rovatunkban foglalkoztunk már vele, hogy a termelési folyamat egyes résztevékenységeit lefedő számítógépes célrendszerek alkalmazása gazdaságosabbá és hatékonyabbá tehető integrált rendszerekké való összekapcsolásukkal. Ám — mint fentebb utaltunk rá — az alkalmazott eltérő belső adatkezelési sémák nem segítik elő az integrálás megvalósítását. Emiatt került a nyolcvanas évek közepén a figyelem középpontjába a különböző CA rendszerek között adatkonvertálással kapcsolatot teremtő szabványos adatcsatolók (interfészek) kutatása és fejlesztése. Az adatinterfészekkel szembeni elvárások nagyok, mivel az önmagukban nagy teljesítményű részrendszerek alkalmazásának eredményességét jelentősen rontja, ha a közöttük lévő adatcsatolás információvesztést okoz, nem kielégítő sebességű és megbízhatóságú, valamint ha többszöri transzformációval, esetleg emberi beavatkozással valósul meg.

A szabványos adatinterfészek megvalósításának elvi és gyakorlati alapját a kettős adattranszformáció és a semleges fájlformátummal való kommunikáció jelenti (3. ábra). Ezt a megoldást a rendszerfejlesztők és a felhasználók többsége elfogadja, sőt sok esetben az egyetlen gyakorlati megvalósítási formának tekinti. A korszerű interfészek tervezése az ISO OSI (open system interconnection) referenciamodell alapján történt, illetve történik, amely a nyitott rendszerek közötti kommunikációt egy hétrétegű struktúra és protokollok szabványosításával segíti elő. A szabványos adatinterfészek a legfelső két réteg formalizálását és feladatait valósítják meg.

Az adatátviteli interfészek között három csoportot lehet képezni:

1. Termékmodell-interfész.
2. Felületmodell-interfész.
3. Általános interfész.

Az első csoportba sorolható az IGES (ANSI — USA), a PDDI/PDES (ANSI — USA), a SET (Aerospatiale — Franciaország), a STEP (ISO — nemzetközi). A másodikba a VDA-FS (DIN — NSZK), az XBF (ANSI — USA) és az AIS (ANSI — USA) tartozik. A harmadikat a CAD*I (ESPRIT — Nyugat-Európa), a DXB (ANSI — USA), valamint a VDA-PS (DIN — NSZK) képviseli. Az említettek közül a nemzetközi viszonylatban közel egy évtizede használt és folyamatosan továbbfejlesztett IGES- (initial graphics exchange specification) előírás játszik meghatározó szerepet. Az IGES alkalmazása mellett szól, hogy: (1) nagyon sok kereskedelmi forgalmazású rendszerbe a hozzá való fordítót beépítették; (2) követi az általánosan elfogadott ASCII információszemléltetési formátumot; (3) a kommunikációs fájlok tartalma könnyen értelmezhető és feldolgozható; (4) alkalmazhatósági területe kiterjed a termékelőállítási folyamat közel minden szakaszára; (5) a fájlok konvertálással és információvesztés nélkül a jövő szabványának tekintett STEP specifikáció szerinti fájlokká alakíthatók át.

Sorozatunk következő részében a széleskörű hazai alkalmazás elősegítése érdekében áttekintjük az IGES termékmodell adatátviteli interfészének tartalmát és a kommunikációs fájl kezelésének főbb sajátosságait.

Horváth Imre

BAREX COMPUTER

SZAKÜZLETNEK

NEM KELL REKLÁM

1054 Budapest,
Bajcsy-Zsilinszky út 54.
Telefon: 131-0946
Tel./fax: 111-6025

INFORMÁCIÓKÉRÉS: 25 ▲

Mégis szalagon?

Bár a számítógépeket általában merevlemezes egységekkel látják el, előfordulhat, hogy adataink, programjaink váratlanul eltűnnek a lemezről. Ezt a merevlemez meghibásodása, de vírusfertőzés is okozhatja. A megelőzés egyszerű: nagyobb gondot kell fordítanunk arra, hogy az adatokat rendszeresen hajlékonylemezre vagy — ha nagy mennyiségű adatról van szó — szalagos tárolóra mentsük.

Miért jó a mágnesszalag?

A mágnesszalagos tárolási mód akkor előnyös, ha rengeteg adatot kell egymás után egyszerre tárolni. A jelenleg forgalomban lévő kisebb teljesítményű mágnesszalagok tárolókapacitása 40 MB. A rendkívül nagy kapacitású, Exabyte gyártmányú forgófejes egységek 5-8 GB-ot tárolnak kazettánként.

Már kaphatók és jól kezelhetők az Insite és a Brier cég által gyártott, igen nagy, 40 MB kapacitású hajlékonylemezes egységek, ám széles körben még nem terjedtek el. Mágnesszalagos egységet sok cég gyárt, a hozzá való kazetták ott állnak a legtöbb, számítógépet forgalmazó bolt polcain. A kazetta pedig ma jóval olcsóbb, mint a nagy kapacitású hajlékonylemez.

Egyvalamiben körözi le a hajlékonylemez a szalagot: a tartósságban. Ugyanis a lemeznek vastagabb az anyaga (fólia), és mágneses rétege is ellenállóbb a külső behatásokkal szemben.

Hogy állnak ebben a versenyben az egyszer írható optikai lemezek, a WORM-ok? Legnagyobb hátrányuk, hogy a meghajtóegységek és a hozzájuk való lemezek nagyon drágák (egység: 4-8 ezer, lemez: 50 dollár).

Bármilyen lemeznek a szalagos egységekkel szembeni kétségtelen előnye akkor jelentkezik, ha a tárolt információ visszakeresésére nem sorban egymás után, hanem tetszőleges módon van szükség. A sávról sávra lépés mindig egyszerűbb és gyorsabb, mint a szalag előre-hátra csévélése. Adatok mentésekor (backup) a mentés-visszatöltés szekvenciálisan, sorban egymás után történik, emiatt a fenti hátrány nem jelentkezik.

Szalagformátumok

A gyakorlatban négy szalagformátumot használnak: a félinches orsóst, a negyedinches kazettást, a 8 mm-es kazettást és a 4 mm-es DAT (digital audio tape) formátumot.

Ezek közül legjobban a félinches (gyakran kilencsávosnak nevezett) orsós szalag terjedt el. Az adatokat párhuzamos sávokon inchenként 800, 1600, 3200 vagy 6250 bitsűrűséggel rögzítik rajta. A sávokat egyszerre olvassa le a fej, ami nagy adatátviteli sebességet jelent, viszonylag kis szalagsebesség mellett. Egy ilyen orsós szalagon általában 145 MB tárolható. Ezek a szalagok nagyon megbízható tárolást nyújtanak, bármelyik egységen biztosan olvashatók, de áruk és aránylag nagy méretük miatt nem célszerűek.

„Okos kazetták"

A szalagos adatmentő rendszerek leggyakoribb tárolóeszköze a negyedinches szalagos kazetta, amely két méretben kapható: a DC2000-es minikazetta és a kicsit nagyobb méretű DC600-as. A kazetták új családja, a DC9135 típusjelű méretben a DC600-ashoz hasonlít, de 900 oersted koercitív erejű szalagot tartalmaz, ami nagyobb tárolási sűrűséget ad. A negyedinches kazetták mechanikája nagyon ötletes; érdekes, hogy sem az audio-, sem a videokazettáknál nem használják, bizonyára szabadalmi okokból. A mechanika lényege egy hajtószalag-kialakítás, amely a szalagot tartalmazó orsókat mindig a megfelelő módon mozgatja és feszíti. A kazetta nem igényel külön orsómeghajtást.

A negyedinches szalagszabványok

A szabványok publikálója a Quarter Inch Cartridge Drive Standards amerikai szervezet (QIC) már 36 szabványt definiált: például a szalagos egységek és a számítógépek közötti interfész, szalagformátumok, a rögzítőfejek tulajdonságai, hibajavító kódok, adattömörítési algoritmusok stb. (Egy külön táblázatban összefoglaltuk a legfontosabb QIC-szabványokat.)

A szabványosítás eredményeként az egyik gyártó meghajtóján felvett szalag lejátszható a másikon, így ipari szabványúvá válhat. (Természetesen minden gyártónak van saját „legjobb", de inkompatibilis formátuma a szabványosak mellett.)

Ez a legegyszerűbb és az egyik leghatásosabb módszer a szalagos tárolók teljesítményének fokozására: sajnos a tömörítő algoritmusok zöme, azok, amelyek nagyfokú tömörítést biztosítanak, csak kb. 64 kB/s sebességgel dolgoznak, ami a szalagos egységeknél nem elég.

A problémát az egyesült államokbeli Stac Electronics cég oldotta meg adattömörítő-visszaállító áramkörének kifejlesztésével, amely 750 kB/s sebességgel végzi az adatátalakítást. Az eredmény egy felére tömörített adathalmaz. A módosított Ziv-Lempel eljárást alkalmazó áramkör mindössze 16 KB RAM-területet használ fel.

„Floppy-szalag"

A QIC-40 és QIC-80 jelű szabványokat nevezik néha így, mert úgy tervezték őket, hogy a rendszerekben már meglévő, a hajlékonylemez-egységet vezérlő áramkörök (floppy-kontrollerek) szalagok írására és olvasására használhatók. Ez a megoldás olcsóbb, mintha külön szalagegység-vezérlő kártyát használnának, és nem foglalnak el külön helyet az amúgy is mindig kevés

I/O csatlakozóból. Ennek a megoldásnak van azonban néhány hátránya.

Először: mivel a legtöbb floppyvezérlő adatátviteli sebessége vagy 0,25 Mbit/s (XT) vagy 0,5 Mbit/s (AT), ezért korlátozott a szalagos átviteli sebesség felső határa.

Másodszor: számos számítógépnél (főleg az AT-knál) két hajlékonylemezmeghajtót terveztek, ezért nincsenek külön kiválasztó vonalak a külön egységekhez. Így a két floppyt használó rendszerekben ügyes szoftver- és hardverötleteket kell alkalmazni, hogy a szalagegység is használhassa az illesztőt. Nagy gond, hogy a két meghajtó elfoglalja a dobozon lévő kivágásokat, így nem jut hely a szalagegységnek. Ilyenkor különálló szalagegységet kell használni, ami már nem igazán olcsó.

A QIC-40 és QIC-80 jelű szabványokban foglaltak szerint a kontroller a szalagon lévő adatokat 1 kilobájtos „diszk-szektorok"-ként látja.

Forgófejes rögzítés

A jelenlegi, kazettánként több mint egy gigabájtot tartalmazó szalagos rendszerek a videomagnóknál ismert forgófejes megoldású rögzítést használják. Ennek az a lényege, hogy a forgófej tengelye a szalaggal néhány fokos szöget zár be. Ilyen módon a forgófej hengerpalástján lévő író-olvasó fejek a szalagot több csíkban olvassák el.

A 4 és 8 mm-es szalagegységek azonban különböznek egymástól. Az Exabyte cég 8 mm-es rendszere a Sony videomagnó mechanizmusát használja; három fej van: a szervo-, az olvasó- és az írás után olvasó fej. Különálló fej törli le a szalagot. A fejhez 221 fokos ívben simuló szalag egy sávja 8 kilobájt tárolására képes, ami kazettánként 2,3 GB tárolókapacitást jelent. A fejlesztések az 5 GB tárolási kapacitás elérésére irányulnak.

A Hewlett-Packard és a Sony által fejlesztett és szabadalmaztatott DDS (digital data storage) formátum 4 mm-es szalagot használ, és a mechanika 4 fejet tartalmaz a forgódobon: két író- és két (írás után) olvasó fejet. A szalagra a sávokat párban írják. A szalag mechanikai igénybevétele kisebb, mert a forgódobhoz csak 90 fokos ívben simul. Tárolási kapacitása jelenleg 1,3 GB.

Más cégek is foglalkoznak hasonló rendszerek fejlesztésével. Ilyen például a Data/DAT elnevezésű rendszer. Ez a DDS-szel sok tekintetben megegyezik, de van egy extra tulajdonsága: az adatokat a helyükön képes újraírni. A DDS támogatói figyelmeztetnek, hogy ilyen rendszernél a gyakori START/STOP miatt a szalag nagy igénybevételnek van kitéve.

A verseny folyik...

A leírtakból látható, hogy jelenleg a forgófejes és a hagyományos egy- vagy többfejes megoldás versenyez egymással. A forgófejes mechanika azzal a megoldással, hogy egy szalagdarabot több sávban tapogat le — és ezzel 38 cm/s szalagsebességet biztosít — nagyon vonzó.

Időközben a QIC bejelentette a QIC-1350-es szabványon alapuló szalagegységét. Ez speciális fejet, nagy koercitív erejű (900 Oe) szalagot, adattömörítő áramkört tartalmaz, és a negyedinches kazettán 1,35 gigabájtot tárol.

Ám lehet, hogy már elkésett, mert időközben számos gyártó megvette a DDS licencét és felkészült a gyártásra.

A jövő versenyt jelent, amelynek hosszabb távon a felhasználók a győztesei, mert olcsóbban, nagyobb tárolási kapacitású szalagegységekhez juthatnak.

A QIC szabványok

Interfészek

QIC-02 Negyedinches szalagmeghajtó intelligens interfész.

QIC-36 Negyedinches szalagmeghajtó alap-interfész.

QIC-104 SCSI alkalmazása a QIC-kompatibilis sorrendi tárolóknál.

QIC-121 SCSI-2 alkalmazása a QIC-kompatibilis sorrendi tárolóknál.

Gyakran használt szalagformátumok:

QIC-24 Sorosan tároló kazetta információcserére (60 MB).

QIC-40 Hajlékonylemez-vezérlő kompatibilis formátum információcserére (40 MB).

QIC-80 Hajlékonylemez-vezérlő kompatibilis formátum információcserére (80 MB).

QIC-120 Sorosan tároló kazetta információcserére (125 MB).

QIC-150 Sorosan tároló kazetta információcserére (150 MB).

QIC-525 Sorosan tároló kazetta információcserére (525 MB).

QIC-1350 Sorosan tároló kazetta információcserére (1,35 GB).

Adattömörítés

QIC-122 Negyedinches szalagmeghajtó adattömörítési formátuma.

Kónya László

A Mikrobazár rovatban rövid, szöveges, a mikroszámítógépekkel kapcsolatos hirdetéseket közlünk. A díjszabás kereskedelmi tevékenységet folytatóknak gépelt soronként (60 karakter) 100 Ft, másoknak az első sor 50 Ft, minden további sor 20 Ft.

Kérjük, hogy a hirdetés díját a Cédrus Rt.-nek a Budapest Banknál vezetett 380-66760 számú számlájára utalják át, vagy postautalványon a Cédrus Rt. címére (1251 Budapest XI., Karolina út 17.) fizessék be, a hátoldalán feltüntetve, hogy apróhirdetés. A befizetést igazoló szelvényt a közlendő hirdetési szöveggel együtt az Alaplap szerkesztőségéhez küldjék el: 1251 Budapest, Pf. 71.

A PC Turbo Klub tagjai ebben a rovatban 20%-os kedvezménnyel hirdethetnek!

ADOK

Amiga 500 1 MB-ra bővítve eladó. Irányár: 53 900 Ft. Cím: Keresztes Gábor, 1142 Budapest XIV., Laky-köz 11. Tel.: 251-2523.

Amigára eladó több mint 2000 lemez játékprogramokkal és felhasználói programokkal. 3,5"-os lemezek 380 Ft-os, 5,25"-os lemezek 750 Ft-os áron eladók. Cím: Keresztes Gábor, 1142 Budapest XIV., Laky-köz 11. Tel: 251-2523.

Eladó **Enterprise-128** számítógép magnóval, joystickkel, 400 db programmal, a Spectrum Világ című újság több példányával. Cím: Hajdú György, 1188 Budapest XVIII., Erdősáv u. 39.

C/16-, C+4-tulajdonosok, figyelem! **Hangdigitalizáló program** megrendelhető kazettán, gyári fejállással, kazettával együtt 250 Ft-ért. Érdeklődők postai utalvánnyal fizessenek, levelet ne küldjenek! Cím: Erős András, 5000 Szolnok, Jászi Ferenc út 22/1.

Eladó **Commodore+4,** magnóval, programkazettákkal, fényceruzával, floppymeghajtóval, lemezekkel, szakirodalommal. Minden megkímélt állapotban van. Ára csak 30 000 Ft! Cím: Éliás Sándor, 3232 Mátrafüred, Pálosvörösmarti út 32.

Eladó **Commodore-64,** floppyvezérlővel, lemezekkel, programokkal. Tel.: Heinbach József, (06-27) 42-600.

Eladó **NISC rendszerű C-64,** 1541-es floppymeghajtóval, Comrex CR-220-as printerrel, fényceruzával, játék- és felhasználói programokkal. Eladó még 2 db mikrokapcsolós és 2 db hagyományos joystick, szakirodalommal együtt. Összesen csak 50 000 Ft-ért! Tel: Hunor Péter, (06-80) 22-091.

Eladó első kézből **Philips CM-8802 RGB-monitor** Eurocart és RCA composit videó (PAL) csatlakozással számítógéphez vagy videóhoz. Ajánlatokat az alábbi címre kérem: Krauss Ottó, 1251 Budapest, Pf. 29.

Eladók **Flight Simulator 4.0**-hez alaposan kibővített, javított térképek, 470 Ft-os áron. Cím: Hódi Gyula, 2170 Aszód, Falujárók útja 5/23.

VESZEK

Keresek **C-64-re** 2.0-ás, vagy bármilyen más verziójú Geos-t, valamint Geos segédprogramokat (lehetőleg angol nyelvűeket). Kapcsolatba lépnék IBM gépen dolgozó, programozási nyelvek iránt érdeklődőkkel! Cím: Szabó Péter, 5100 Jászberény, Korányi út 1.

CSERÉLEK

Eladó **Enterprise-128,** magnóval, joystickkel, 500 db programmal és szakirodalommal. **IBM XT**-hez winchestert veszek, programot cserélek. Cím: Gulcsik István, 5000 Szolnok, Széchenyi u. 5. II./8.

INFORMÁCIÓKÉRÉS: 11 ▶

RENDSZERVÁLTÁS
A SZÁMÍTÁSTECHNIKÁBAN!

A MÚLT
Eddig PC-k (XT-től — 486-ig), alkatrészek, perifériák, ajánlatok és egyéb kiegészítők forgalmazásával foglalkoztunk.

A JELEN
Most mindezeket hálózatba kötve, telepítve, bevizsgálva TPA és jogtiszta DEC rendszerekbe is integráljuk.

3 M TERMÉKEK

Floppy lemezek
Streamer kazetták
Mágnesszalagok

SZÁLLÍTÁS RAKTÁRRÓL
AZ ÉRDEKLŐDŐKET VÁRJUK IRODÁNKBAN
ÉS RÖVIDESEN ELKÉSZÜLŐ BOLTUNKBAN!

Kérje részletes árlistánkat!

MACRODA

MACRODA KERESKEDELMI KFT.

1016 Szírtes u. 28/a
Telefon: 186-5782, 186-5686, 185-7866
Telefax: 186-5686 Telex: 22-5375

INFORMÁCIÓKÉRÉS: 12 ▶

NYÁK KFT EXPRESSZ

KERESKEDELMI IRODA
1046 Budapest IV., Pamutgyár u. 3.

ELEKTRONIKAI ALKATRÉSZEK

Számítástechnikai gyártók, szervizek figyelmébe ajánljuk szolgáltatásunkat.
Egyszerűbb lesz anyagbeszerzése, ha

MINDENT EGY HELYEN

nálunk rendel meg.
Belföldi és import alkatrészek rendelése:

TELEFONON, TELEFAXON: 169-3320

Ütemezett gyártáshoz, ütemezett szállítás. Mennyiségi árlépcsők.
Processzorok, memóriák, interfész, csatlakozók, kábelek.
Ipari elektronika, SMD technika is. Kurrensebb alkatrészek már

KÉT HÉT ALATT IS,

különleges alkatrészek rendelésre.

AKCIÓ!! Reklám-áron...



GRAPHIPLOT digitális vezérlésű dobplotter

A készülék szolgáltatásai:

— Minden funkcióra kiterjedő öntesztelés.
— Üres rajzpapírt befogadó, szabadon futó henger.
— GP–01M tollváltós típusnál a 8 db toll beszáradásmentes tárolása a megfogókban.

ÁRAINK:

A/0:	180 000 Ft + Áfa
A/1:	160 000 Ft + Áfa
A/3, A/4:	39 600 Ft + Áfa

VISINFORM színkontrasztos kijelzőcsalád

Számjegy-, betű- és mátrixkijelző



Egyedi megvalósítás:

— Utastájékoztatók repülőtereken, vasúti és autóbusz-pályaudvarokon.
— Hirdetőtáblák színes és egyszínű grafikus kivitelben.
— Ügyfélszámhívók bankokban, orvosi rendelőkben.
— Közlekedésirányító táblák autópályákon, közlekedési csomópontokban.

a FOKGYEM-től

FINOMMECHANIKAI ÉS ELEKTRONIKUS MŰSZERGYÁRTÓ SZÖVETKEZET

1775 Budapest XXII., Nagytétényi út 100–102.
Telefon: 173-0011. Telex: 22-60-34.

FOK GYEM

INFORMÁCIÓKÉRÉS: 24 ▲

Szóval sysop szeretnél lenni?

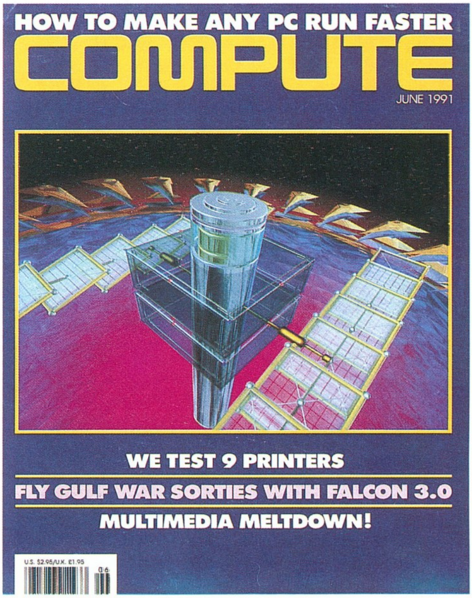
Akik hébe-hóba felhívnak egy BBS-t, valószínűleg eltűnődnek azon, milyen is lenne egyszer egy saját rendszert üzemeltetni. Sok időbe, nem kis erőfeszítésbe, türelembe és pénzbe is kerül, hogy valaki sysoppá válhasson, de előnyeit sem szabad véka alá rejteni. (Sysop = system operator = rendszerüzemeltető.) Nézzük most meg, mit is jelent felállítani egy saját BBS-t (BBS = bulletin board system = elektronikus hirdetőtábla, információközvetítő rendszer).

Komputered és egy modem segítségével a világ bármelyik BBS-ét felhívhatod a normál telefondíjszabás árán. Saját BBS-eddel viszont a szó szoros értelmében az egész komputeres világgal beszélgethetsz. Rendszeredet használhatod programok, állományok cseréjére, de hangot adhatsz bármivel kapcsolatos véleményednek is. A BBS nem más, mint egyfajta kommunikációs forma közted, a sysop és a többi felhasználó között. Nem lényegtelen az sem, hogy eszmecserétek valóságos legyen, hisz a BBS lényege éppen az, hogy a megszerzett információkat megosszad a felhasználók és komputerrajongók között, pontosan úgy, mint például egy élménybeszámoló beszélgetésben.

A rendszer kiépítése előtt érdemes egy felhasználói csoporthoz csatlakoznod, ahol rengeteg jó tanácsot kaphatsz, de élőben is tanulmányozhatod a BBS működését, olcsón vagy akár ingyen juthatsz hozzá első programállományaidhoz. Saját rendszeredet ugyanis komoly szoftverarzenállal kell indítanod úgy, hogy már a kezdet kezdetén is nyújthass valamit hívóidnak. Ezek után azt is el kell döntened, hogy privát (csak a baráti köröd számára elérhető) vagy nyilvános rendszert kívánsz-e kiépíteni. Egy szűk körű rendszer üzemeltetése már alkalmas lehet arra, hogy felmérjed, megfelelő sysop-anyagból gyúrtak-e. Ha úgy találod, hogy igen, akkor érdemes rendszeredet saját telefonvonaladról egy másik — bérelt — vonalra költöztetned, és egyben nyilvánossá is tenned.

A BBS-sel nem fogsz meggazdagodni! Ha csak alapvető szolgáltatásokat tartasz a gépeden, még a telefonszámlád sem lesz túl borsos, de megállapodhatsz hívóiddal a költségek megosztásában is.

De nézzük most meg, mi szükséges még egy átlagos BBS üzemeltetéséhez! Elég lesz egy 64 K-s, esetleg 128 K-s számítógép, egy floppymeghajtó-egység, egy 300 bps átviteli sebességű modem és esetleg egy nyomtató. Ilyen — viszonylag kis — kiépítettségnél az állományok méretét limitálnod kell ugyan, de alapvető tapasztalataidat már egy ilyen konfiguráció mellett is gyarapíthatod. Gyakran kell majd a lemezeket cserélgetned, felhasználóidnak pedig be kell érniük azzal az anyaggal, ami aktuálisan a meghajtódon forog. Állományaid feltöltése megint más kérdés. A célod ugyanis az, hogy te is minél több olyan programot kapj, amit hívóiddal megoszthatsz. A kis kiépítettségű rendszer viszont csak kevés program egyidejű fogadására alkalmas, ezért ügyelned kell a lemezek cseréjére, a rendszert állandóan gondoznod és felügyelned kell! Ideális lenne tehát saját BBS-edet egy 20 MB-os vagy nagyobb kapacitású winchesterrel is kiegészítened. Így az üzemeltetés szinte problémamentessé válik, kevés figyelmet igényel, ugyanakkor hatalmas mennyiségű szoftvert tarthatsz raktáron, és akkor még nem beszéltünk a merevlemez és a 1541-es meghajtók közötti sebességkülönbségről.



Mielőtt a BBS-edhez szoftvert vásárolnál, szerezd meg az elérhető szakirodalmat, ahol ezek szolgáltatásait áttekintheted. Ne add fel addig, míg az általad megálmodott rendszerhez a legmegfelelőbb programcsomagot meg nem szerzed. Más szoftver szükséges egy párbeszéd-orientált BBS-hez, más a fájl- és programcserén alapulóhoz és megint más, ha online játékokat játszotok.

Ne utólag próbálj meg rájönni, hogy kezdetben mit is akartál! Ezért nem árt papír és ceruza segítségével megtervezned, meddig is fog terjedni BBS-ed, elképzelned leendő menüstruktúrádat — a beszerzett dokumentációra is támaszkodva. Ezzel egy csomó időt és fejfájást takaríthatsz meg.

Ha a BBS-t nyilvánossá teszed, attól még megkövetelheted, hogy a felhasználók azonosítsák magukat, netán jelszót használjanak. Minden sysop szereti tudni, hogy ki hívja a rendszerét. Ez a módszer segít a komputer-kalózkodás ellen, ugyanakkor megvéded felhasználóid személyes üzeneteit, elektronikus levelesládáját. Ha nagyon kukacos vagy, megkérheted potenciális hívóidat, hogy BBS-ed első használata előtt postázzák neked igazi nevüket és címüket. Általában nem tanácsos, hogy rendszeredet a hívók számára teljesen hozzáférhetővé tedd. Kivételt képezhetnek azok, akiket nagy biztonsággal tudtál azonosítani. Ha például éppen számítógépednél ülsz, amikor valaki felhívja BBS-edet, normál telefonodon hívd fel az általa megadott számot! A vonalnak foglaltnak kell lennie! Ha nem, az felettébb gyanús, és további nyomozást igényel.

Némi védelmet kell kiépítened a „bőbeszédű" felhasználók ellen is. Nem árt, ha a hívásonkénti időt 45 percre korlátozod, illetve ha egy számról naponta maximum három hívást fogadsz el. Ha ezt nem teszed, néhány felbátorodott hívó szinte kivonja vonaladat a forgalomból, míg mások hiába próbálnak hívni és elbizonytalanodnak. De a sysop is értetlen, ha nem hívják a rendszerét. Egy verőfényes nyári napon

például ne számíts csúcsforgalomra! Ha kevés a hívód, hirdesd meg magad más BBS-eken, komputerújságokban vagy fénymásolt szórólapokon, amiket könyvtárakban, iskolákban vagy számítógépboltokban adhatsz le.

Ha mindezek után is úgy érzed, hogy alkalmas lennél sysop-nak, akkor vágj bele! Ehhez segíteni fog saját BBS-ed gyümölcseinek betakarítása. Nagyon sokat tanulhatsz meg a komputerekről, egy csomó szoftvert gyűjthetsz össze hívóidtól, és emberekkel társaloghatsz a világ minden tájáról. Ráadásul senkit nem kell hívnod, ők keresnek meg téged! Szívet melengető érzés felhasználóid elismerő véleménye rendszeredről — és ezen keresztül rólad is.

(Compute, 1991/március)

A Sun Devil hadművelet

Amerikában a törvény emberei úgy szállnak szembe a számítógépes bűnözéssel, hogy közben csak halvány sejtéseik vannak a vonatkozó jogszabályok szándékait és alkalmazási technikáját illetően. A Harvard Egyetem Jogi Karának egykori hallgatója, Buck BloomBecker rajtuk, illetve a számítógépes kalózkodás áldozatain próbál segíteni jó egy évtizede.

BloomBecker — tízéves közvádlói múlttal a háta mögött — 1979-ben megalapította az NCCCD-t (National Center for Computer Crime Data). Célja az volt, hogy az USA összes felderített számítógépes bűnügyének jogi körülményeit és az eljárások részleteit adatbázisba gyűjtsék, segítve ezzel a precedens ügyekben eljáró jogászokat. Ugyanakkor nem hivatalos kapcsolatokat építettek ki adatvédelmi szolgálatokkal, könyvvizsgálókkal és rendszeroperátorokkal is. Mivel az információt igénylő személy egyben adatszolgáltató is, az állomány naprakésznek tekinthető, ami azért fontos, mert az USA államaiban évente tekintélyes mennyiségű új törvényt alkotnak a komputer-kalózkodás vagy éppen a vírusok terjedése ellen.

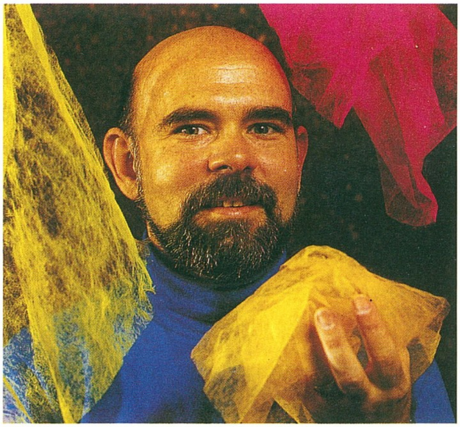
A hatékonynak szánt jogi támogatás azonban visszafelé is elsülhet, ezért BloomBecker óva int a túlbuzgóságtól. Ám a törvény cerberusai — fittyet hányva az intelemre — megindították a Sun Devil hadműveletet. Az akció 1990 májusában fejeződött be, és kétéves titkosszolgálati nyomozás előzte meg, amelybe 150 szövetségi ügynököt és számos helyi, valamint állami jogi szervezetet vontak be. A hadjárat krónikája 27 letartóztatásról, jó néhány BBS bezárásáról, 40 számítógép és 23 000 mágneslemez elkobzásáról számol be. Az akciónak köszönhetően viszont megalakult az EFF (Electronic Frontier Foundation), amely már nemcsak a törvénycikkek gyűjtésére, hanem azok korrekt és jogszerű alkalmazására is koncentrál. A Sun Devil hadművelet ugyanis számos alkotmányozási és polgárjogi kérdést vetett fel.

A társadalom egy része az akciót annak látta csak, amit a hatóságok is igyekeztek láttatni velük, azaz hajtóvadászat egy falka ifjú csibész ellen, akik a „sötétség légiójának" nevezik magukat.

Mások viszont azt nehezményezték, hogy az ügyben érintett hackereket sötét és gonosz bűnözőkként festették le, holott vétkük mindössze annyi, hogy felfedezték és kihasználták a „cyberspace"-t. (Ez William Gibson sci-fi író terminológiája, közelítő magyar megnevezése „kibernetikai hézag" lehet, és a számítógépes rendszerek eddig még nem kutatott, feltérképezetlen és ismeretlen mezsgyéjére utal.)

Jogi szempontból viszont a Sun Devil hadművelet a baklövések sorozatának bizonyult. Betiltották például a Phrack nevű komputeres újságot, amit pedig az alkotmány ugyanúgy véd, mint a New York Timest. Volt olyan komputer-kalóz, akit még óvadék ellenében sem helyeztek szabadlábra, mondván, hogy ő minden idők legelvetemültebb számítógépes bűnözője. Hónapokra betiltották a Steven Jackson-féle játékokat csupán annak a gyanúnak az alapján, hogy egyik alkalmazottjuk — egy BBS sysop (rendszergazda) — esetleg szintén komputer-kalóz. Egy másik BBS sysop esetében pedig az a gyanú merült fel, hogy a Texasi Egyetem számítógépét használta fel üzelmeire. Mégsem az egyetemen jelent meg házkutatási paranccsal az FBI, hanem a gyanúsított lakásán, és az egyetem komputere helyett a lakáson található személyi számítógépet foglalták le.

Az akció hátterét vizsgálva sok hasznos tanulság vonható le. A számítógépes rendszerbetörést — amit a hackerek szívesebben hívnak rendszerfeltörésnek — kezdetben azért övezte a társadalom viszonylagos toleranciája, mert éppen időszerű volt. Elődje, a telefonbetyárkodás a 60-as évek Amerikájában szintén megtűrt lázadási mód volt olyan monolit egységek ellen, mint a gigászi telefontársaságok. A számítógépes rendszerek liberalizálásakor még éltek ezek az attitűdök, és alternatívaként a komputer-kalózkodás irányába hatottak. Ma tulajdonképpen a számítógépes bűnözés már a hagyományos bűnözési műfajok között (lopás, csalás) is jelen van, a komputerrendszerek széles körű alkalmazása miatt. Mi hát a megoldás?



BloomBecker szerint a felhasználókat sokkal felelősebb komputerezésre kell szorítani, az EFF álláspontja viszont a teljes liberalizációt és felvilágosítást célozza. Ez ellen BloomBecker a maga híres példáját hozza fel: az Iran-gate ügyben inkriminált North ezredes egy rendszerbeli hurkot használt fel arra, hogy ellenőrizhetetlen beszélgetést folytasson Poindexter admirálissal. „Egy kicsivel több szakértelem, és sohasem tudtuk volna meg, miben is sántikálnak" — mondja szkeptikusan. A technikai robbanás eredményeként a törvény emberei úgy érezték, hogy elvesztették uralmukat a dolgok irányítása felett. Ennek azután természetes velejárója volt, hogy enyhén kétes módszerekkel próbálták visszabirkózni a hatalmat a naprakész hackerektől. A Sun Devil hadművelet ennek a filozófiának volt az első „éles" próbája.

BloomBecker szerint, ha a törvény a sztereotíp kalóztípusra koncentrál — intelligens, fiatal férfira —, akkor a biztonsági rendszerek csődöt mondanak. A felelős komputerezés

mellett a törvény embereit is meg kell tanítani a szakma trükkjeire, mert „addig, amíg egy komputer-kalóz ügyében eljáró FBI-nyomozó nem tud egy ROM-chipet egy satutól megkülönböztetni, nem várható érdemi fordulat" — kesereg.

A komputer- és kommunikációs rendszerek tükrében újra kellene gondolnunk az adatvédelem elemi szabályait, és végre meg kellene húzni azt a határvonalat is, amin túl az adatszerzés már bűncselekmény. BloomBecker szerint ezeket a dilemmákat Amerikában inkább az EFF tudja majd feloldani — presztízse, erőforrásai és hatékony eszköztára révén —, semmint a küzdelmet elindító NCCCD, amelyik kellő erő hiányában csak asszisztált egy szélmalomharchoz.

(Personal Computer World, 1991/március)

Az önszervező rendszerek

A számítástechnika három olyan nagy problémakörrel küzd, amelyhez teljes szemléletváltás szükséges, mert az eddig bevált programozási, gép- és rendszerépítési módszerek ezeken a területeken nem alkalmazhatók.

1. Alak- és hangfelismerés.
2. Az egész Földet behálózó rendszer.
3. Párhuzamos felépítésű számítógépek.

A számítástudomány most tart ott, hogy megérti, vannak olyan feladatok, amelyeket szinte lehetetlen programozni, ezért megpróbáljuk a természettől kölcsönkérni azokat az alapelveket, amelyek alapján például az alak- és hangfelismerés történik. Valószínűleg ez az a fő irány, amerre a filozófia, a matematika és a programozástechnika tart, hogy jobban megérthessük a körülöttünk lévő világot. Teljesen persze a programozást sem dobjuk el, mert vannak dolgok, amelyeket az idegrendszer nagyon rossz hatásfokkal végez. Így például azt, hogy 3+2=5, legjobb esetben is több százmillió sejtünk számítja ki, vagy mondjuk nyelvünket több ezer évig sok millió ember csiszolta a maga sok milliárd agysejtjével, mire ilyen lett.

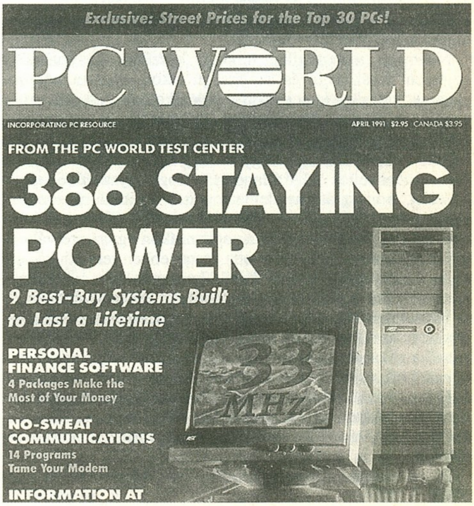
A molekulákat, atomokat, elemi részecskéket ilyen sorrendben egyre kevésbé ismerjük; nem tudjuk, miből van egy elemi részecske. A molekulák sejteket, a sejtek embereket, az emberek egy-egy ország társadalmát, ezek pedig a földi civilizációt alkotják. Számomra itt a tréfás csomag legkülső doboza. Ebben a pillanatban szinte hallom, hogy tiltakoznak az ufó-hívők, miszerint a Föld igenis kölcsönhatásban áll más civilizációkkal. Egy százalék esélyt adok nekik, de valahol meg kell húznom a vonalat. Az élő sejt minden egyes molekulája élettelen. Ezek szintén élettelen, de működő kémiai automatákat alkotnak. E kémiai rendszerek közötti önszerveződés teremti az életet. Két érv küzd egymással: az egyik azt mondja, hogy egy sejt annyira bonyolult, hogy teremtéséhez Isten (Sátán, Életerő, Varázslat stb.) szükségeltetik, mert ennyi bonyolult molekula soha nem gyűlik össze egy helyre véletlenül; a másik azt, hogy egyszerű gázokból aminosavakat, ezekből fehérjékhez hasonló anyagot, ezekből meg néhány más anyagból osztódó és saját anyaguk egy részét szintetizáló képződményeket hozhatunk létre. Mindkét csapat megfeledkezik az élő és az élettelen közötti szerveződési szintről.

A többsejtű, önálló mozgásra képes élőlények azért alkalmazkodnak jól környezetükhöz, mert rendkívül gyorsan reagálnak annak változásaira, és ezt idegrendszerüknek köszönhetik. Az idegrendszer, ha csak néhány sejtből áll is, óriási előnyt jelent azokkal szemben, amelyeknek ilyen sincs. Egy idegsejt utasításkészlete rendkívül csekély. Bemeneti oldalán súlyozottan összegzi pozitív előjellel az ingerületeket, negatívval a gátlásokat, és ha az összeg pozitív irányban meghalad egy kritikus szintet, kimenetén megjelenik egy impulzus. Ez után bizonyos holtidő telik el a következő kimeneti impulzusig, még akkor is, ha a bemeneti jel változatlan. Ez a holtidő helyettesíti a központi órajelet, és lehetővé teszi a rendszer összehangolódását, bár oka eredetileg az, hogy a sejt csak szakaszosan képes energiát leadni. A bemenetek súlyozása a tanulási folyamat során alakul ki, ez és a sejtek viszonylag egyszerű huzalozása helyettesíti a programozást.

Az, hogy a sejtek embereket építhetnek fel, nyilvánvaló, mint ahogy az is, hogy az embert mint fogalmat nem a sejtek építik fel, hanem a sejtek közötti kapcsolatok. Az emberi értelem viszont nem az egyes emberek tulajdonsága, hanem a társadalomé. Az egyes országok társadalma viszont szemünk láttára szerveződik valami nagyobb és bonyolultabb szervezetté. A legutóbbi háború jó példa erre: egy erősebb ország megtámadott egy gyengébbet, mire az összes többi megdöbbentően rövid idő alatt kőkeményen keresztbe tett az agresszornak. Ha az ENSZ-nek sikerül egyre inkább az egész emberiség érdekeit képviselnie, mindent felülmúló hatalommá válik, és mindaddig, amíg ezzel vissza nem él, irányíthatja a Földet. Ehhez tengernyi észre és bölcsességre lesz szüksége.

Létezik tehát egy rendezőelv, amely egyszerű elemek összekapcsolásával összetettebb rendszereket hoz létre. Az önszerveződés lényege, hogy az alkotóelemek úgy állnak kölcsönhatásban egymással és a környezetükkel, hogy elérhessenek egy bizonyos egyensúlyt. Ma egyetlen ilyen folyamatot tudunk pontosan leírni: a kristálynövesztést sóoldatból. Ekkor molekulák kaotikus kavargásából rendezett struktúra jön létre. Persze nem mondhatjuk, hogy az atomok AKARNAK elérni egy bizonyos állapotot, így a definíciót is fejre kell állítani: azt a dinamikus egyensúlyi állapotot nevezzük önszervező struktúrának, amelyik… És most önmagával nem magyarázhatjuk. Mi itt a gond ?

Hát az, hogy egy ilyen szervezet látszólag teljesen ellentmond az entrópia elvének, miszerint minden magára hagyott zárt rendszer a rendezetlenebb állapota felé törekszik. Ismét ez a fránya törekszik…, akar…, szeretne… Mondjuk inkább



azt, hogy egy magára hagyott zárt rendszerben nagyobb a valószínűsége a sűrűség kiegyenlítődésének, mint a különböző sűrűségű tartományok elkülönülésének — energiáról vagy bármilyen közegről legyen szó. No mármost az atomoknak nem kell akarniuk semmit, és az ellentmondás is részben fel van oldva. Különben sem kell nekünk az entrópia; egy egyensúlyi zárt rendszerben nem jönnek létre önszervező rendszerek. Kristályt is úgy növesztünk, hogy az a tértartomány, ahol a rendezett struktúra létrejön, soha nem zárt és soha nem egyensúlyi; mindig anyag és energia áramlik át rajta. Létezik tehát a természetben egy olyan törvény, ami látszólag az entrópia ellen dolgozik, de valójában kiegészíti azt: miközben egy szinten kiegyenlítődés történik, a kölcsönható elemek létrehozhatnak egy olyan összetett rendszert, amelynek stabilitása és a környezetével fennálló dinamikus egyensúlya belső állapotváltozásainak körfolyamata révén marad fenn.

Ez a dolog egyik oldala. A másik: gondolatkísérlet. Vegyünk egy nagy számítógépmemóriát, töltsük fel 1 és 0 bitekkel véletlenszerűen, majd tegyünk bele egy pici programot, amely:

1. Keres egy helyet, önmagán kívül.
2. Másolatot készít önmagáról azon a helyen.
3. Elindítja a másolatot.
4. Tovább működik.

Eredmény: először lassan, majd egyre gyorsabban rendezett tartományok jönnek létre. Ezután a dolog háromesélyes:

1. A folyamat valahol látszólag vagy ténylegesen leáll.
2. A rendezett és rendezetlen állapotok véletlenszerűen kavarognak.
3. Létrejön az események zárt körfolyamata.

Jellemző még a modellre az, hogy a kezdeti állapotok parányi változása szélsőségesen felerősödik. (Matematikusok azt mondják: divergál.) Mivel géppel dolgozunk, az említett véletlen természetesen nem igaz, de a zárt körfolyamat ismétlődési hossza vagy a stabil állapothoz vezető eseménysorozat hossza már 64K esetén is iszonyúan nagy szám.

Ezt a háromféle viselkedést már W. Ross Ashby homeostatja is tudta. Ez analóg gép volt, négy összekapcsolt szervorendszerből állt, és az idegsejthálózat viselkedését akarták modellezni vele. A közismert Életjáték Neumann sejtautomatáinak legegyszerűbb változata.

Neumann matematikai modellt készített: gyakorlatilag végtelen hosszúságú és szélességű síkot, amelyet négyzet alakú, egymáshoz kapcsolódó sejtek alkotnak. A sejtek struktúrája, az állapotváltozás-függvény minden sejtre azonos, előre adott és időben állandó. Ez azt határozza meg, hogy a sejt következő állapota hogyan függ előző állapotától és a szomszédos sejtek előző állapotától. Hogy egyszerűbb legyen a dolog, a vizsgálat kezdetén a sejtek az általunk vizsgált sejtcsoportok kivételével úgynevezett nulla állapotban vannak, és a sejtek állapotukat az egész síkon egyszerre változtatják. Neumann bebizonyította, hogy ebben tervezhető egy olyan sejtcsoport, amely a ciklikus működés különleges eseteként önmagáról működő másolatot képes készíteni.

Modellezhető tehát az élőlények egyik legfontosabb tulajdonsága: az önreprodukció. Egy egysejtű persze nemcsak ezt csinálja. Szabályozott anyag-, és energiacserét folytat környezetével; saját apróbb sérüléseit képes kijavítani; elhasználódott részeit újakra cseréli; a környezet lassú változásaihoz bizonyos határok között alkalmazkodik. A többsejtűek ezenkívül információt dolgoznak fel, és megváltoztatják környezetüket.

A számítástechnika nagy eredménye, hogy létrejöttek az óriási kapacitású, olcsó háttértárak, és azok az eszközök (lézer, üvegszál), amelyek lehetővé teszik nagy mennyiségű információ gyors, nagy távolságú átvitelét. Ezzel megjelent a lehetőség és vele együtt az igény is az egységes világméretű számítógéphálózatra. Egy ekkora hálózatot egy központból vezérelni azért lehetetlen, mert mindig a központ lenne a szűk keresztmetszet és inkább akadályozná, mint segítené a hálózat részei közötti kapcsolatteremtést. Jobb, ha meg sem kíséreljük egy ilyen rendszer üzembe helyezését, mert utólag óriási munka lenne a műszaki és anyagi csődtömeg felszámolása, és az átállás egy valóban hatékonyan működő új rendszerre. Most kell elvileg jól megtervezni az egészet, és beleálmodni az összes előre nem látható bővítési lehetőséget is, mert utólag változtatni egy ekkora méretű és komplexitású rendszeren nem lehet.

Hogyan lehet megtervezni az előre nem láthatót? Úgy, ahogy az élőlények idegrendszere felépül: néhány egyszerű alapelv, keresési, adatszűrő, azonosító és tanuló algoritmus; minden teljesen alárendelve a célnak, hogy az objektum alkalmazkodni tudjon a környezetéhez és a hozzá hasonló objektumokhoz. Az emberi környezet felé az illesztőfelület lehet egyedi, de a hálózat felé egységesnek kell lennie. A gép lehet bármilyen, ha kapacitása és sebessége elegendő. A vonalillesztőnek is csak néhány alapvető fizikai és forgalmazási paramétert kell tudnia. A vezérlőprogram bonyolultsága viszont minden eddigit felülmúl, mert önmaga másodpéldányaival tökéletes biztonsággal, olajozottan kell együttműködnie. Ez lehet egy 3 usque 20 megabájtos program, amely annál kisebb lehet, minél jobban szakítani tudunk eddigi programozási szemléletünkkel. Ennek a programnak ugyanis harmadfokú önszervező rendszernek kell lennie. Az értelmes lények társadalma csak 40-50 féle atomból épül fel, mégis értelmes, mert minimum ötödfokú önszervező rendszer (ahol DNS=1, riboszóma=2, neuron=3, bogár=4). Tehát egy ilyen program bonyolultsága szinte eléri a legegyszerűbb bacilusét. Ezt létrehozni titáni méretű és jelentőségű feladat.

A számítástechnika harmadik, gyorsan fejlődő sötét területe a párhuzamos feldolgozásnak nevezett biztató kísérletezés. Ezekkel a gépekkel olyan folyamatokat modellezhetünk majd, amelyeket nem tudunk vagy nem lehet előre elemi utasításokra lebontani. Ha összekapcsolunk akár kettő, akár több mint egymillió mini- vagy mikrogépet, a szűk keresztmetszet itt is a működésüket összehangoló program és a csatlakozópontjaikat összekötő drót lesz. Az előbb említett, önmagához hasonlóval tökéletesen kommunikáló programot nem integrálhatjuk egy parányi elemi adatfeldolgozóba. Mi a megoldás? Először is el kell dönteni, hogy mire akarjuk használni ezt a gépet. A felépítés nem független a feladattól. Fel kell mérni, hogy egy elemi adatfeldolgozás milyen bonyolultságú, és ennek megfelelően lehet megtervezni az alapelem méretét (memóriaméret és utasításkészlet). Nagyon gondosan kell megtervezni az alapelemek közötti adat- és utasításforgalmat, és ennek minél nagyobb részét integrálni az alapelembe. Továbbá: az elemek összekapcsolódását úgy kell megtervezni, hogy ha az egyik elem feldobja a talpát, az adatutak kikerülhessék, és az egész rendszer működőképes maradjon. Gondoskodni kell arról is, hogy a textúra minden irányban akármekkorára bővíthető legyen, egy 64 cellás gépet könnyedén lehessen 256-osra bővíteni. Meg lehet nézni, hogy mi az elvi különbség a 8 x 8-as négyzetrács és a henger- vagy gömbfelületre tett 64 cella között. Külön problémakör az, hogyan juttatom be az adatokat és veszem ki az eredményt. Célszerűbb itt is minden egyes elemhez eljuttatni az információ egy-egy részét, mert a textúra szélén lévő kapcsolódási pontokat további bővítésre lehet felhasználni.

(PC World, 1991/március)

Hódít a laptop

Amíg a nagyok technikai és árháborújukat vívják a piacon, addig a kutatók igyekeznek előre látni a fejleményeket. Egyes elemzők szerint a közeljövőben a legnagyobb eladási felfutást a laptop és notebook kategóriájú gépek fogják elérni. A hordozható mindentudók népszerűsége érthető, hiszen általuk lakásunktól és munkahelyünktől távol tartózkodva sem kell lemondanunk egy nagy teljesítményű munkaeszközről. Fizikai méretük egyre zsugorodik, egyre apróbbak és könnyebbek. Az elemek sem merülnek már le olyan gyorsan és könnyebbek is lettek.

A technika fejlődésével elképesztő lehetőségek nyílnak meg. A japán NEC cég például nemrég olyan géppel jelent meg, amely mindössze három kilogramm tömegű, és összekapcsolható a számítógéprendszerekkel – mégpedig rádióhullámok segítségével. A rádiótelefont is közbeiktatva a felhasználók fantasztikus mozgásszabadságot kapnak. Üzletemberek, menedzserek repülőről, autóból vagy akár a tárgyalóteremből, értekezlet alatt hívhatják saját számítógépes rendszerüket, vállalatuk központi gépét.

Ennek ellenére a laptop-készülékek sajnos még mindig számos kompromisszumra kényszerülnek, főleg ami a képernyőt illeti. (Bár az IBM azt hangoztatja szlogenjében, hogy „egy laptop kompromisszumok nélkül".) Az is tény, hogy ezek a gépek sokkal kisebbek már nem lehetnek, mert kezelésükhöz bizonyos hozzáférhetőségre, kézméretnyi helyre szükség van. A továbblépéshez már a hanggal vezérelhető és kézírásfelismerő, igazi „jegyzetfüzet-számítógépek" tökéletesítése szükséges.

Számítógépes elsősegély

A számítógépes szolgáltatások piacán dinamikusan fejlődik a katasztrófaelhárítás üzletága. Mióta a hangsúly eltolódott a „vasról" az adatra, egyre fontosabb lett az információ védelme. Nemcsak az adatokhoz való illetéktelen hozzáférés megakadályozása tartozik ide, hanem az adatok elvesztésének kiküszöbölése is. Hazánkban ahány cég, annyiféle helyi szabályzat írja elő a mentések rendjét, az adatlemezek tárolását. A duplikált winchestertől a páncélszekrényig minden előfordul a palettán!

Az ICL (International Computers) angliai vállalat, amelyben a világhírű japán Fujitsu cégnek többségi részvénye van, és az ugyancsak brit Sherwood új vegyes vállalatot hozott létre számítógépes vészhelyzetben nyújtandó szolgáltatásokra. Ez azt jelenti, hogy például valamilyen természeti katasztrófa esetén (tűz, földrengés, árvíz stb.) a megsemmisült vagy sérült számítógépes rendszereket azonnal kisegítik egy adatfeldolgozó központtal, szigorúan üzleti alapon.

Multiprocesszoros NCR

Processzorok tömeges összekapcsolásán alapuló számítógéprendszer forgalmazását kezdi meg várhatóan 1991. szeptemberében az NCR. Ennek a rendszernek legfőbb erőssége, hogy olcsó mikroáramköröket használ fel, amelyek egymás teljesítményét megsokszorozva rendkívül nagy műveleti sebesség elérését teszik lehetővé. Eddig ilyen rendszereket csak műszaki-tudományos kísérletek, számítások céljára alkalmaztak.

A hamarosan forgalomba kerülő 3600-as gépcsalád lesz az első, melyet ügyviteli rendszerként való felhasználásra ajánlanak. Az NCR az amerikai számítógépgyártó piac ötödik legnagyobb vállalata. Ausztriai székhelyű európai képviselete 1992-től tervezi a magyar piac kiszolgálását.

Keleti vírusveszedelem

Nem nindig örülnek Nyugaton a kelet-európai változásoknak! Nyugat-Európa adatfeldolgozó rendszereit ugyanis tömegével lepték el a „vasfüggöny mögül kiszabadult", új vírusok. A keleti programozók — ha már piaci részesedésüket egyelőre nem nagyon tudják növelni — leleményességükkel „kápráztatják el" nyugati kollégáikat. Például olyasmival, hogy a vírus a saját hosszát levonja a program új hosszából, és ezt a változatlanságot imitáló fájlhosszt tárolja. Másik galád trükk a vírushordozó vírus, amelyet viszonylag könnyű felfedezni, de annak „kiirtásakor" megszületik a rejtettebb másik vírus.

Sok kicsi sokra... menne

A Balaton Trade Kft. által a közelmúltban szervezett kiállítás alapgondolatát az adta, hogy a szaporodó új vállalkozások egyre-másra keresik az egyszerűbb, kisebb számítógépes rendszereket és a célfeladatokat megoldó pénzügyi, ügyviteli stb. programokat. Főleg olyan komplett rendszerekre lenne igény, amelyek nem túl nagy összegért (100-200 ezer forint) könnyen kezelhető gépet és programokat tartalmaznak.

A kiállító cégek (a C&C Szentendréről, a SZÜV Lícium Rt., a SZÜV Metakód Leányvállalat, a Balaton Elektronika Kft. és a Kvalitatív Kft. Kecskemétről) éppen ezeket az olcsó komplett rendszereket, rövid programokat kínálták.

A hannoveri CeBIT-en külön szekciója van a kisvállalkozásokat segítő számítástechnikának. A hazai első fecskét is igazán követhetné a többi, a Compfairen és az Ifabón sem feledkezve meg a legdinamikusabban fejlődő vállalkozástípus „komputerizálásáról".

A sakk ördöge

Abból az alkalomból, hogy a Softinvest megkezdte a Mephisto sakk-számítógépek magyarországi forgalmazását, ismét vitatéma lett, hogy legyőzhetik-e valaha is a számítógépek a legjobb sakkozókat. Az élversenyzők néhány évvel ezelőtt még határozott nemmel válaszoltak, most azonban már kezdenek meginogni. A idei hannoveri CeBIT-en rendezett versenyen bemutatkozó Deep Thought-2 például megverte a német bajnokot. Ez a gép ugyan multiprocesszoros különlegesség, 2500 Élő-pontszám körüli játékerővel, de a szériában készülő Mephisto család legerősebb tagjai, amelyek a legutóbbi világbajnokságot nyerték (Lyon), ugyancsak tekintélyes, 2350-es pontszámúak. Kaszparov és Karpov is azok közé tartoznak, akik korábban nem tartották valószínűnek az emberi sakktudást felülmúló számítógépeket. Véleményük változóban van.

Nagy hal az akváriumban

Sokan feltették a kérdést a Computer 2000 magyarországi megjelenésekor, hogy a tőkeerős, kizárólag nagykereskedelmi tevékenységet folytató cég tevékenysége milyen változásokat eredményezhet a magyar piacon. A széles áruválaszték, a nagy készletekből garantált azonnali szállítás révén igen fontos szerephez juthatnak a hazai viszonteladók ellátásában. Ráadásul a gyártóktól kapott kedvezmények miatt sok esetben alacsonyabb áron hozhatják forgalomba a számítástechnikai termékeket, mint amennyiért azokat a dealerek akár a termelőtől közvetlenül megvásárolhatják. A kis halak talán aggódhatnak a nagy hal láttán, de a vásárlók inkább örülnek a fejleményeknek. Legalábbis reménykednek az árakat lefelé szorító valódi versenyben.

Tiszai Tibor

Utrecht is a ringben

EUROPE SOFTWARE '91
INTERNATIONAL COMPUTER SOFTWARE EXHIBITION
INTERNATIONALE AUTOMATISERINGSBEURS

Nem sokkal a CeBIT befejezése után Bécsben, majd Budapesten is lezajlott az IFABO. A budapesti szakvásáron szinte minden valamirevaló, számítástechnikával foglalkozó magyar cég képviseltette magát, a veterának számító, több éve a piacon lévő cégektől kezdve az újonnan alakult legkisebbekig. A magyar standokon többnyire a hardver uralta a helyet, s ez inkább a kereskedői, semmint az alkotó szellem előretörését jelezte. A bevált szoftverek mellett csak az ügyviteli rendszerek igazolták, hogy mindig képesek a megújulásra. Május végén — a párizsi SICOB-nak a „nagyok" által kikényszerített elmaradásával — a hollandiai Utrecht látta vendégül a világ élvonalát a Europe Software '91 kiállításon. Megnéztük, mennyire kötelez a név.

Az IFABO-n tapasztaltak alapján nem is kerestük a magyar szoftverszerzőket a kiállítók között, mégis találtunk egy fecskét. Az egyetlen magyar stand a Daten-Kontor és a Metrimpex közös fellépésének volt köszönhető. Ezen a standon egy fogorvosi rendszer, a miskolci *microCAD*-en látott Dental keltett nagy érdeklődést. Utrechtbe a program német nyelvű változatát hozták el, csakúgy, mint az ugyancsak német megrendelésre készült konferenciarendszert, amellyel az előadók regisztrálásától, szállásfoglalásától kezdve a teljes költség- és honoráriumelszámolás elvégezhető.

A kiállításon három szekció különült el, a robotika és az általános szoftverek békés egymásmellettiségben a Julianahalban, míg a LANWORLD (a hálózatok világa) teljesen elkülönítve a Margriethalban mutatkozott be.

A robotikának otthont adó folyosón nagy feltűnést — és olykor riadalmat — keltett az önmagától közlekedő szállítókocsi, mely külső érzékelőivel tájékozódott (nem lefektetett huzal alapján), minden akadályt kikerülve. Eddig ehhez hasonló szállítóeszközökkel inkább csak raktárakban, nagyobb gyárakban találkozhattunk, de ott a pálya előre meghatározott.

Egy másik helyen a robotkarok rendkívül finom mozgásokra történő programozása tűnt fel: a robot játszi könnyedséggel birkózott meg például az almahámozással.

A hagyományos, általános felhasználói szoftvereket a néhány nagy szoftverház mellett jobbára a hollandiai viszont- és viszont-viszonteladók képviselték, az igazi nagy nevek inkább a LANWORLD-öt választották. Az Ashton-Tate előadássorozatban mutatta be „legújabb" termékeit (dBASE IV 1.1...). A WordPerfect valóságos nonstop show-t rendezett szoftverei tiszteletére. (Bár a szoftverük lenne fele olyan jó, mint a marketingjük, nálunk is bízvást népszerűbb lenne.) Sajnos igazából egyik óriás sem mutatott be semmi újat. A kiállítóterem középső részét az itthon rettegve figyelt Computer 2000 foglalta el. A CAD rendszerek királya, az AutoCAD 11 Windows-os ruhában is megmutatkozott. Minden itt megjelent reklám fennen hirdette a király dicsőségét, míg a háttérben megbúvó konkurens RoboCAD a maga csendes módján ugyanerre a trónra pályázik. A jelek szerint egyelőre még kevés eséllyel.

Először üdvözölhettük nagy nyilvánosság előtt a Windows nagy riválisát, a GeoWorks programrendszert. A GeoWorks hatalmas előnye a böhönc Windows-zal szemben a kis memóriaigény (XT-n is minden további nélkül fut), valamint a talán még egyszerűbb kezelhetőség. Remélhetőleg rövidesen itthon is megjelenik, vállalkozó kedvű forgal-

Demólesen

Az újságírók a kiállításokon mindig hódolnak olthatatlan gyűjtőszenvedélyüknek. Az Alaplap munkatársának pedig a kötelezően begyűjtendő labdák, kalapok, nadrágtartók stb. mellé „munkaköri kötelességként" szert kell(ene) tennie minden fellelhető demólemezre, hogy annak figyelemre méltó tartalmát közkinccsé tehesse a lap mágneslemez mellékletén. Így volt ez Utrechtben is. Zsákmányunkat azután kíváncsian dugtuk a gépbe a „magyar" standon. Az eredmény lehangoló volt. Az egyik lemez — egy flamand vírusdetektor — elindulni még csak hajlandó volt, de tartalma messze elmaradt a nálunk kapható, hasonló programok teljesítményétől. A Progress demója pedig maga volt a demókészítő kisiparos szegénységi bizonyítványa, méltatlan a cég egyébként körültekintő, színvonalas reklámtevékenységéhez. Mintha megéreztük volna, hogy valami gond lesz, két példányt is beszereztünk belőle. Hiába. Az összes fellelhető gép meg az összes trükk kevés volt ahhoz, hogy sikerüljön elővarázsolni a lemez tartalmát. Nem jártunk sokkal jobban a többi ajándéklemezzel sem: pénztárgép-konfigurációnk nem lévén, egy professzionális kivitelű demóval sem tudtunk mit kezdeni, amelyikbe pedig sikerült bepillantanunk, az alatta maradt a mércének. Így sajnos e havi lemezmellékletünkre nem kerülhetett „madárlátta" szoftver.

V. J.



mazókban talán nincs hiány. Hogy egy kicsit gyorsítsunk az ügymeneten, íme a gyártó GeoWorks címe: 2150 Shattuck Ave., Berkeley, CA 94704. Tel.: (415) 644-0083.

A Fox cég régen beharangozott új terméke, a FoxPro 2.0 sajnos nem jelent meg a kiállításon (pedig Magyarországon már kapható). A Data Access cég műhelyében készült Dataflex legújabb verziója viszont várhatóan nagy sikert arat majd nálunk is. A Dataflex 2.3 teljes SQL-környezettel, önálló editorral, nagyon egyszerűen kezelhető alkalmazásgenerátorokkal felvértezve jelent meg. Sajnos a termék önnállóan futtatható (EXE) állományt nem fordít. Jobbára itthoni kiállításokhoz szokott látogatókként nagy meglepetéssel vettük észre — többek között — itt, a Data Access standján, hogy a kiállítók igen magas szinten értenek ahhoz a portékához, amit kiállítottak! Ismervén a magyar piac felvevőképességét, széles körű elterjedésre nem is a Dataflex rendszer számíthat, hanem a FlexQL, amely nevéhez híven a fájlformátumokat tekintve rendkívül rugalmasan igazodik a felhasználási környezethez. A „mindent menüből" alapállás a FlexQL beszámoló-készítőt a felhasználók körében különösen népszerűvé teheti.

És még egy magyar vonatkozású hír. Lehetséges, hogy az évről évre listavezető Recognitának az optikai karakterfelismerő rendszerek között elfoglalt trónját az amerikai Calera cég WordScan rendszere foglalja el a közeljövőben. A Recognita eddigi külföldi népszerűségének egyik tényezője a nagyon alacsony ár volt, most viszont a tudásban legalábbis egyenrangú WordScan már árban is versenyképes, így élesebbé válhat a verseny.

Különleges hardvermegoldásoknak elvileg nem lett volna helyük az Europe Software-en, mégis örülhettünk, hogy ilyeneket is láttunk. Elég csak a Fast cég Screen Machine rendszerét említeni. A vezérlőszoftver Windows 3.0 alatt fut, bármilyen alkalmazásra ablakot lehet vele nyitni, ahol valós idejű digitalizált kép bemutatása a cél. A kép helyzete változtatható, más ablakok bármikor átfedhetik. A rendszerhez egy speciális kártyára, valamint a Windows alatt működő szoftverre van szükség. A videokamera és a digitalizálókártya segítségével winchesteren rögzített filmet bármikor vissza lehet játszani. A képeket a rendszer a lemezen tömörítve tárolja.

Minden valamirevaló hardvergyártó rendelkezik pénztárgépekkel, ezekből a nélkülözhetetlen eszközökből is jutott néhány a kiállításra. A 286-os AT-nek speciális 265 billentyűből álló klaviatúrája van, a normál 101 gombon kívül minden egyes billentyű külön címezhető. A kisméretű monitor (Carry) a számlakészítő programnak teljesen megfelel. Hasonló eszközt már Magyarországon is forgalmaznak.

A LANWORLD kiállítás — mint már említettük — a nagyok találkozóhelye volt. Az összes UNIX, XENIX, NOVELL... képviseltette magát. Mindenki valami régi-új termékkel próbálkozott. A NOVELL bemutatta az új 3.00 sorozatot, a Microsoft a Windows alatt dolgozó LAN Manager programját, mely leginkább az Object Vision rendszerre hasonlít.

Végül is mennyire felelt meg a rendezvény a nevének? Hát inkább volt Software, mint Europe. Sajnos a kiállítást valahogy nem nemzetközi közönségre tervezték, s ez a csak holland nyelven kiadott szórólapokon erősen érződött. Ugyanakkor dicséri is a kiállítókat, hogy az angol nyelvű információt kérő névjegyeket nem szuvenírnek tekintették: mire hazaértünk, jó néhány cég már postán el is juttatta hozzánk termékismertetőit. A számítástechnikai túltermelés mindenesetre érezhető volt, akárcsak a magyar IFABO kiállításon. Leggyakrabban itt is a kiállítók vizsgálták egymás termékeit.

Csiki András

Összefogás egy humánus ügyért

A Netcom Kft. a közelmúltban felhívással fordult a számítástechnikai vállalatokhoz, hogy támogassák a Lares Alapítványt! Az alapítvány óvodájában olyan épelméjű, de súlyos beilleszkedési, alkalmazkodási és kommunikációs zavarokkal küzdő, autisztikus gyerekek gyógyító nevelésével foglalkoznak, akik a hagyományos oktatási rendszerből kiszorultak. Az ingyenes óvoda mostoha körülmények között működik, adományokból tartja el magát.

Mindennapi munkát szinte lehetetlenné tevő elhelyezési gondok enyhítésére a Svéd Ház Kft. egy épület felépítését ajánlotta fel. A házhoz tartozó telek megvásárlásához szükséges pénzösszeg azonban csak újabb adományokból teremthető elő.

Ezen szeretne segíteni a felhívásával a Netcom Kft. Ezért kezdetben 100 000 Ft-ot ajánlottak fel erre a célra és egyúttal felhívták a számítástechnikai cégeket, hogy ilyen humánus célokra fordítsanak többet bevételeikből, járuljanak hozzá a Lares Alapítványhoz!

A GEM operációs rendszer XII.
Metafüggvénytan

Az előző alkalommal a GEM vektorgrafikus metafájl ismertetését kezdtük el, most a metafájlban alkalmazható VDI (virtual device interface) függvényekkel folytatjuk a sorozatot.

A GEM metafájlban elsődlegesen vektorgrafikát tárolhatunk, de lehetőségünk van IMG formátumú bittérképes rasztergrafikus kép tárolására is. Ebben az esetben a metafájl fejrésze 14. szavának, a rasztergrafika flagnek az értéke 1. Ez azt jelenti, hogy a metafájl tartalmaz rasztergrafikát, pontosabban azt, hogy a metafájlban szerepel egy v_bit_image-függvényhívás.

A függvénycsoportok jellemzői

A szabványos metafájl-függvények körébe tartoznak a rajzoló, az attribútum-beállító, az általános és speciális célú meg a fájlkezelő függvények. A metafájl kezelésére is függvényeket kell használnunk, amelyek szintén a VDI-könyvtárhoz tartoznak.

A rajzolófüggvényekkel vonalat, téglalapot, lekerekített sarkú téglalapot, körívet, kört, körcikket, ellipszist és ellipsziscikket rajzolhatunk. Kiszínezett objektumokat is megjeleníthetünk: például téglalapot, lekerekített sarkú téglalapot, sokszöget, kört, körcikket, ellipszist és ellipsziscikket. Az előbb felsorolt függvények vagy a rendszer alapértelmezése szerinti, vagy az attribútum-beállító függvényekkel előzőleg beállított attribútumokat használják.

Az attribútum-beállító függvényekkel határozhatjuk meg a rajzfüggvények által használt attribútumokat. Ezektől függ a rajzfüggvények által megrajzolt alakzatok vonaltípusa, vonalvastagsága, színe, a kitöltőminták, a kitöltési mód, a kitöltőminták színe.

Szín, minta

Velük állíthatjuk be a kiírandó szövegek attribútumait is: a betűtípust, a karakterszélességet, a szöveg színét és elforgatásának szögét. Ezekkel a függvényekkel szabályozhatjuk a különféle szövegkiemelési effektusokat is, például a vastagságot, aláhúzást, árnyékolást, dőlt betűt stb. A felsorolt effektusoknak bármely kombinációja is használható. Szintén ebben a függvénycsoportban találjuk azokat a funkciókat, amelyekkel beállíthatjuk egy vonal elejének és végének a lezárását. Választhatjuk például azt, hogy a vonal ne simán, hanem legömbölyítve vagy nyílban végződjön. Standard vonalak és kitöltőminták használatán kívül lehetőség van arra is, hogy a felhasználó adjon vonaltípust és definiáljon új kitöltőmintát.

Az általános célú függvények segítségével törölhetjük a munkaállomást, a pufferban lévő grafikai utasításokat azonnal végrehajthatjuk, levághatunk egy megadott téglalapon kívül eső területet.

Nyomtatás

A speciális függvények az output eszközt vezérlik. Kinyomtatják a megadott szöveget, az IMG formátumú fájlt, törlik a nyomtató pufferét, lapot dobnak a nyomtatón, szöveges módból grafikusba állítják vissza a nyomtatót vagy a képernyőt. E függvények segítségével nyomtathatjuk ki egy kép koordinátákkal meghatározott, téglalap alakú területét.

A fájlkezelő függvényekkel inicializálhatjuk a puffert, kitölthetjük a headert, bejegyezhetünk néhány függvényhívást. Ezeket a függvényeket kell használnunk akkor is, amikor bejegyezzük a képméretet a headerbe, elmentjük a metafájl pufferének tartalmát, lezárjuk a fájlt.

Most pedig nézzük meg, hogy az előbb felsorolt csoportosítás szerint milyen függvényeket használhatunk az egyes csoportokban! A függvénynév után zárójelben az operációs kód és tőle kötőjellel elválasztva az alkód található hexadecimális számrendszerben.

Rajzolófüggvények:

v_pline ($6), v_pmarker ($7), v_gtext ($8), v_fillarea (9), v_bar ($B-1), v_arc ($B-2), v_pieslice ($B-3), v_circle ($B-4), v_ellipse ($B-5), v_ellarc ($B-6), v_ellpie ($B-7), v_rbox ($B-8), v_rfbox ($B-9), v_justified ($B-$A), v_recfl ($72).

Attribútum-beállító függvények:

vst_height ($C), vst_rotation ($D), vs_color ($E), vsl_type ($F), vsl_width ($10), vsl_color ($11), vsm_type ($12), vsm_height ($13), vsm_color ($14), vst_font ($15), vst_color ($16), vsf_interior ($17), vsf_style ($18), vsf_color ($19), vswr_mode ($20), vst_alignment ($27), vsf_perimeter ($68), vst_effects ($6A), vst_point ($6B), vsl_ends ($6C), vst_updat ($70), vsl_udsty ($71).

Általános célú függvények:

v_clrwk ($3), v_updwk ($4), vs_clip ($81)

Speciális célú függvények:

v_exit_cur ($5-2), v_enter_cur ($5-3), v_form_adv ($5- $14), v_output_window ($5-$15), v_clear_disp_list ($5-$16), v_bit_image ($5-$17), v_alpha_text ($5-$19).

Fájlkezelő függvények:

v_ponwk ($1), v_meta_exetns ($5-$62), v_write_mtea ($5- $63), vm_filename ($5-$64), v_clswk ($2).

A következő alkalommal elkezdjük a VDI függvények részletes ismertetését.

Kovács P. Attila

115 200 baudos soros vonal

Galoppból vágtába

Amikor a DOS-leírásban a MODE COM#: parancsról először olvastam, és a beállítható sebességeknél 9600 baudnál nagyobb értéket nem láttam, azt gondoltam, hogy biztosan csak elírás. Nehezen tudtam elképzelni, hogy — mivel a termináloknál már általánossá vált a 19 200 és a 38 400 baudos sebesség — a PC nem biztosítja ezeket. Egy idő után belenyugodtam: 9600 baud a legnagyobb sebesség, és ezért a nyomtatót, a plottert, az EPROM-égetőt ehhez az alacsony sebességhez igazítottam. Természetesen két PC összekötésére, és hosszabb fájlok átküldésére gondolni sem mertem. Csak amikor a vonalat 115 200 baud sebességgel használó kommunikációs programokkal találkoztam, kezdtem el vizsgálni a soros vonal lehetőségeit.

Áttekintés a poroszkálás ritmusában

Legmagasabb szinten a soros vonalat egy speciális fájlként (COMx) kezelhetjük. Ezt bármely programozási nyelven a szokásos fájlkezelési műveletekkel megnyithatjuk, írhatjuk és olvashatjuk. A DOS fájlkezelő parancsai is használhatók. Egy fájlt a

COPY FILE COM1

paranccsal küldhetünk el az egyes soros vonalra.

A vonali kommunikáció formátumát a MODE COMx:... DOS paranccsal állíthatjuk be. Ez azonban csak az 1. és 2. soros vonalra, és csak 9600 baud sebességig működik.

A következő szint a BIOS soros vonal kezelő 14H-es interruptja. Ennek 4 alfunkciója van:

AH=0: Vonal beállítása.

AL bitjei határozzák meg a formátumot és a sebességet:

0-1	A karakterhossz kódja: 10 — 7 bites, 11 — 8 bites.
2	A stopbitek száma: 0 — egy stopbit, 1 — két stopbit.
3-4	A paritás meghatározása: 00 — nincs paritásbit, 01 — páratlan paritás, 10 — páros paritás.
5-7	Bitsebesség: 000 — 110 baud 001 — 150 baud 010 — 300 baud 011 — 600 baud 100 — 1200 baud 101 — 2400 baud 110 — 4800 baud 111 — 9600 baud

DX a kiválasztott vonal sorszáma (0-3).

AH=1: Karakter küldése.
Az elküldendő karaktert AL-ben kell megadni.
DX a kiválasztott vonal sorszáma (0-3).
Hiba esetén AH0 hibakódot ad vissza.

AH=2: Karakter beolvasása.
DX a kiválasztott vonal sorszáma (0-3).
A vett karakter AL-be kerül. Hiba esetén AH0 hibakódot ad vissza.

AH=3: A vonal állapotának lekérdezése.
DX a kiválasztott vonal sorszáma (0-3).

AX bitjei a vonal állapotát adják:

15	Timeout lejárt.
14	Adási regiszter üres.
13	Karakter adásra kész.
12	Vonali break jött.
11	Formátumhiba.
10	Paritáshiba.
9	Vevő túlfutás.
8	Vett adat van.
7	DCD modemjel.
6	RI modemjel.
5	DSR modemjel.
4	CTS modemjel.
3	DCD változott.
2	RI változott.
1	DSR változott.
0	CTS változott.

A funkciókból látszik, hogy a BIOS-on keresztül elvégezhető minden olyan sorosvonali művelet, amely nem igényli a vonal interruptos kezelését és 9600 baudnál nagyobb sebesség beállítását.

Átlépve a határon

9600 baudnál nagyobb sebesség beállítása csak a sorosvonali csatoló átprogramozásával valósítható meg. Most nem célunk a 8050-es soros illesztő összes regiszterét részletesen ismertetni, mivel azokra csak egész speciális esetekben van szükség. (A részletes leírás megtalálható Pethő Ádám: IBM PC/XT felhasználóknak és programozóknak 3. A ROM BIOS és ami mögötte van. Számalk, Budapest, 1989. VI. 3. fejezet.) Az itt leírt rutinnal átprogramozott csatoló a BIOS-on vagy akár a DOS-on keresztül jól használható. Ezt

> PC-be különböző gyártójú sorosvonali csatolókártyákat használhatunk. Ezek legtöbbje lehetővé teszi a nagy sebességű kommunikációt, minden változtatás nélkül. Vannak azonban olyan kártyák, ahol a zavarszűrés javítására a kimenetekre szűrőelemeket tettek (pl. eredeti IBM soros csatoló!). Ezek a nagy sebességű átvitelt megakadályozhatják. Általában elmondható minden csatolóra, hogy a maximális 115 200 baud használata csak rövid (max. pár méter) kábellel eredményes.

a rutint, a BIOS INT 18H/0 helyett kell meghívni. A paraméterezés eltérő, de lehetőséget ad a csatoló teljes kihasználására. A beállítható maximális sebesség 115 200 baud.

A rutin a regiszterek elmentése után a 8250-es sorosvonali csatoló Line control regiszterébe beállítja a Divisor latch address bitet, ezzel engedélyezve az új osztó betöltését. A következő két out utasítás az osztót állítja, majd az utolsó out utasítás a formátumot határozza meg.

Még egyszerűbben használható a nagy sebességű soros vonal C nyelvű programmal. Ezt a DOS MODE COMx:... parancsa helyett kell meghívni. Paraméterezése is nagyon hasonló:

CMODE COM#[:]baud[,[parity][,[data][,[stop]]]]

ahol

- \# A soros vonal száma 1-4.
- baud Sebesség: 50, 75, 110, 135, 150, 300, 600, 1200, 1800, 2000, 2400, 3600, 4800, 7200, 9600, 19 200, 38 400, 57 600, 115 200
- parity A paritás: N — nincs, O — páros, E — páratlan, 0, 1. (A default: E)
- data Az adatbitek száma 5-8. (A default 7)
- stop A stopbitek száma 1-2. (A default 2, 110 baud esetén, 1, egyébként)

A program jelentős része a paramétersor feldolgozásával foglalkozik. Ezek működése a Turbo C debuggerével jól követhető. Egyedül az strcode rutin érdemel némi figyelmet. Ez egy nevet keres egy táblázatban, jelen esetben a baud rate konstanst. Biztosítja a rövidítések felismerését is. Hasznos lehet minden olyan programban, ahol meghatározott kulcsszavakat kell felismerni.

A paramétersor karaktereit a getcc( ) szubrutin végzi. Érdekessége, hogy a C startup program által szétbontott paramétersort összeilleszti, majdnem úgy, mint ahogy azt a DOS eredetileg biztosította.

A vonal programozását ténylegesen a főprogram legvégén található 4 outb( ) utasítás végzi, ugyanúgy, mint az assembler-programnál.

Jó vágtázást!

Pintér Gábor

A programozási nyelvek világa

Olvasóink közül sokan nem számítástechnikusok, de munkájuk során használnak számítógépet, és mint érdeklődő amatőröknek megfordult már a fejükben, hogy a rutin jellegű munkán túl — már csak úgy kedvtelésből is — valamilyen módon „szót kellene érteniük" a komputerrel. A megfelelő „társalgási nyelv" kiválasztásához szeretnénk segítséget nyújtani ezzel az új sorozattal, amelyben a legfontosabb és legelterjedtebb programozási nyelvekből adunk ízelítőt. Igyekszünk majd az egyes nyelveket ismertető részeket az adott nyelv IBM PC-s megvalósításait bemutató és összehasonlító cikkekkel, illetve (a „kóstolást" segítendő) a mágneslemez mellékleten példaprogramokkal kibővíteni.

A számítógép nagyon sok áramkörből álló elektronikus gép. Fizikai megjelenése (a burok és az elektronikus alkatrészek sokasága) a hardver. A tiszta hardver, a csupasz gép a gyakorlatban teljesen használhatatlan, és a feladatoknak a gép számára érthető prezentálásáig még hosszú út vezet. Ezeknek a lépéseknek a nagy részét a számítógép az ember számára szükséges idő töredéke alatt képes elvégezni. De itt rögtön egy paradoxonnal találkozunk: számítógépet akarunk használni, ám a szükséges műveletek egy része éppen abból ered, hogy a számítógépet használhatóvá kell tennünk. A paradoxon feloldását a számítógépek operációs rendszerei és a különböző segédprogramok (editor, fordító, linker) teszik lehetővé. Segítségükkel a programozó, illetve a felhasználó előtt álló problémák megoldhatók, „csupán" a felmerült feladatot kell a számítógép számára érthető módon megfogalmazni. Tehát ahhoz, hogy a számítógép lehetőségeit maximálisan kihasználjuk, a számítógépet programozni kell.

Ritka kivétellel a programokat azért írják (helyesebben tervezik), hogy egy számítógép értelmezze azokat. Ez után a számítógép elvégzi az adott feladatot, azaz a műveletek sorozatát hajtja végre a megadott specifikációk alapján. A program tulajdonképpen szöveg. Mivel általában egy meglehetősen komplex feladatot kell specifikálni, mégpedig a lehető legpontosabban, a legapróbb részletekig, ezért ennek a szövegnek a jelentését nagyon precízen kell meghatározni. A kellő pontossághoz egy egzakt formalizmus szükséges. Ez a formalizmus a (programozási) nyelv. A programozási nyelv feladata a strukturálatlan emberi gondolatok, azaz a fantázia és a számítógép működéséhez elengedhetetlenül szükséges precizitás között húzódó szakadék áthidalása. A magas szintű programozási nyelveket úgy alakították ki, hogy a program megírása egyszerűbb legyen, és a probléma megoldását a nyelv segítségével természetesebb formában lehessen megadni. Ezeknek a szempontoknak a teljesítéséhez a nyelvnek kell megfelelő elemeket biztosítani.

Az emberek közötti kapcsolat a leghatásosabban a nyelv segítségével valósítható meg. A nyelv lehetővé teszi gondolatok és ötletek kifejezését: nélküle az emberi érintkezés nagyon nehéz lenne. A számítógép-programozásnál a programozási nyelv a problémával küzdő és azt megoldani akaró személy, a megoldáshoz segítségül felhasznált számítógép közötti kommunikáció eszközéül szolgál. A nyelv befolyásolja az emberek gondolkodásmódját. Ez egyaránt igaz a beszélt nyelvekre, a matematika nyelvére és a programozási nyelvekre. Azt mondják, hogy a francia költői nyelv, a német mérnöki, az angol pedig ezek keveréke. Az eszkimóknak

nagy szókincsük van a hóval kapcsolatos fogalmakra. A matematika világából, ha a differenciál- és integrálszámítás nyelvét vesszük példának, azt állapíthatjuk meg, hogy bár mind Newton, mind Leibniz egymástól függetlenül kidolgozták az alapelveket, mégis Leibniz jelölésmódja (nyelve) bizonyult érthetőbbnek és használhatóbbnak Newtonénál. Ezért mind a mai napig Leibniz jelölésmódját használják a világon mindenütt. Bár mind Newton, mind Leibniz módszerével megoldható ugyanaz a probléma, Leibniz nyelve mégis könnyebbé tette a megoldást. A programozás terén ugyanez a helyzet. Egy hatékony programozási nyelv fokozza a számítógépes programok írását és fejlesztését, formálja a programozó gondolkodásmódját, és a nyelv minősége nagyban befolyásolja a segítségével létrehozott programok minőségét.

A számítástechnika hőskorában a programozás meglehetősen ijesztő feladat volt. A korai számítógépek közül nagyon sok ún. „hard wired" volt, hogy egy adott feladatot ellássanak. Ahhoz, hogy a programot megváltoztassák, a komponenseket át kellett huzalozni. Abban az időben a forrasztópáka volt a szoftvermérnök legfőbb szerszáma. Neumann és Turing voltak az elsők, akik felvetették a tárolt program elvét, azaz, hogy a program utasításait a számítógép memóriájában tárolják az adatokkal együtt. De nemcsak a programtárolás koncepciója változott, hanem a programok létrehozásának eszközei is. Megindultak a fejlesztések, amelyek célja olyan programozási nyelvek létrehozása volt, amelyek elvezettek a számítógép-specifikus gépi kódú utasításoktól és az assembly-nyelvtől (az ún. alacsony szintű, azaz gép/hardver-orientált programozástól) a magasabb szintű, problémaorientált programozási nyelvek felé. Az első tárolt program elvén működő számítógépeket binárisan kódolt utasítások segítségével programozták. A számítógéppel való párbeszéd természetesen unalmas volt, nagy figyelmet követelt, és mint minden ilyen munka esetében, gyakran fordultak elő hibák, amelyek kijavítása sok időt vett igénybe.

Az első általános célú problémaorientált programozási nyelvek egyike a FORTRAN (FORmula TRANslator = képlet fordító) volt, amelyet 1954-ben John Backus és munkatársai hoztak létre tudományos számítási feladatok megoldására. Mivel a számítógépek csak a gépi nyelvet „értik meg", ezért szükségessé vált olyan programok kifejlesztése, amelyek a FORTRAN-ról gépi kódba ültették át a programokat (fordítók). A FORTRAN nagy sikert aratott tudományos körökben, eszközül szolgált a problémaorientált programozási nyelvek létjogosultságának bizonyítására. Ezt hamarosan újabb nyelvek követték.

1958-ban indult meg a tranzisztoros, azaz az ún. „második generációs" számítógépek gyártása. A számítástechnikai szakértők egyébként ebben az időben kezdték el használni a a hardver és szoftver kifejezéseket. Az új számítógépeket már az operációs rendszerből és többnyire fordítókból álló szoftverrel adták. Az amerikai kormány 1959-ben, főképp üzleti körök nyomására, összehozott egy számítógép-felhasználókból és -gyártókból álló csoportot, hogy tervezzenek egy üzleti programozási nyelvet. A nyelv 1961-ben jelent meg, és COBOL-nak (COmmon Business Oriented Language = üzleti alkalmazásra orientált nyelv) hívták. A COBOL nem rendelkezett a FORTRAN matematikai lehetőségeivel, de a FORTRAN-ban is meglévő tömb struktúrán felül hierarchikus rekordokat is kezelt. A vezérlési utasítások jobban strukturáltak (IF...ELSE, PERFORM...UNTIL), és a fájl kezelése nagyon fejlett.

A FORTRAN és a COBOL azonban nem elégítették ki a számítástechnikával foglalkozó, kérlelhetetlen kutatókat, akik 1960-ban új nyelvet terveztek, az ALGOLT (ALGOrithmic Language = algoritmikus nyelv), a modern blokk struktúrájú nyelvek ősét. Ez a nyelv több, azóta alapvető fontosságú fogalmat vezetett be: például rekurzió, paraméterátadási módok, lokális változók. Sajnos az ALGOL nem terjedt el széles körben. Egy másik irányzat képviselőjeként, John McCarthy a bostoni MIT-en egy listafeldolgozáson alapuló nyelvet tervezett, a LISP-et (LISt Processing). Ez a nyelv a mesterséges intelligenciával kapcsolatos kutatásokra irányult, és mind a mai napig népszerű, különösen a szakértői rendszerek terén. Nagy befolyással volt más nyelvekre is, például a PROLOG-ra, amelyet a japánok ötödik generációs számítógépeikhez választottak.

Az IBM, a világ legnagyobb számítógépgyártója természetesen akkor is a saját útját járta, és egymást követően két programozási nyelvet fejlesztett ki: az APL-t, egy nagyon tömör jelölésmódot használó nyelvet és a PL/1-et. Ez utóbbit arra szánták, hogy a FORTRAN, ALGOL és COBOL előnyeit egyesítse. Ezenfelül a PL/1 két új elvet is magában foglalt: a kivételkezelést és a multitaszkos eszközöket. Igazán népszerűvé csak az IBM gépeken vált. Nagyon komplett, de részben éppen ezért bonyolult nyelv volt. A másik végletet a Dortmouth College-i John Kemény és Thomas Kurtz nagyon gyorsan sikeressé váló nyelve, a Basic jelentette. Nem nyújtott új nyelvi koncepciókat, de ez volt az első valóban interaktív nyelv, és mint ilyen, az integrált fejlesztői környezet révén használata könnyen elsajátítható volt. A világ programozóinak zöme, különösen nagygépes környezetben, mind a mai napig ezen nyelvek valamelyikének továbbfejlesztett változatával programoz, bár az utóbbi időben a trónkövetelők egyre erősebbek lettek.

Villányi László

(Folytatjuk)

Egy titokzatos magazin!

A megmagyarázhatatlan jelenségek, azonosítatlan repülő tárgyak, a jövő- és űrkutatás, a parapszichológia rendkívül érdekes területeivel foglalkozik a kéthavonta megjelenő **Ufómagazin**.

Az Ufómagazin 1991/4. számának tartalmából:

UFO világkongresszus Arizonában

Halálos titok — Gravitációs erővonalak

Lüktető diszkosz — Egy szigorúan ellenőrzött jelentésből

A csodatevő Agykontroll

Kérdések Cooper őrmesterhez — Keviczky Kálmán írása

INFORMÁCIÓKÉRÉS: 26 ▼

Típusok a Modula-2-ben

Az előző részben láttuk, hogy a Modula-2-ben használat előtt minden változót deklarálni kell, vagyis nevük a program deklarációs részében szerepel. A deklaráció szerepe kettős: egyrészt a fordító kiszűri a hibás azonosítókat, másrészt a deklarációval minden változóhoz egy adattípust rendelünk. Az adattípus az adott változóra vonatkozó permanens információ. Egy adott típusnak deklarált változó rendelkezik a típusra vonatkozó minden tulajdonsággal, egészen addig, amíg meg nem szűnik. A típusra vonatkozó információ lehetővé teszi további programtervezési és programkódolási hibák kiszűrését a program forráskódjából, a program működésének elemzése nélkül.

Egy típus az értékek egy halmazát definiálja, vagyis meghatározza, hogy a deklarált változók milyen értékeket vehetnek fel a program futása során, és milyen műveleteket végezhetünk velük. Minden változónak egy típusa lehet, ezt a deklaráció egyértelműen kijelöli. Minden művelet meghatározott típusú operátorokra értelmezett, és az eredmény típusa is meghatározott. Így a forráskódból megállapíthatjuk, hogy melyik műveletet hajtjuk végre egy változóval.

Adattípusokat a programban is deklarálhatunk. Ezek az úgynevezett összetett típusok, amelyek általában alaptípusokból állnak. A Modula-2 több előre definiált elemi típust tartalmaz. Ezek az úgynevezett standard vagy alapvető adattípusok a nyelv részei, és így nem kell őket külön deklarálni.

Természetesen nemcsak a változók rendelkeznek típussal, hanem az állandók, a függvények, az operandusok és a műveletek eredményei is. Az állandók esetében a típus általában az állandó jelöléséből következik, de az állandó típusát explicit módon is deklarálhatjuk. Minden más esetben a típusdeklaráció explicit.

A típusdeklaráció meghatározza, hogy az adott adattípussal rendelkező változók az értékek mely halmazát vehetik fel, ez a deklaráció a típushoz egy azonosítót rendel. A típus lehet egy másik típus azonosítója is. Ha egy típusdefiníció deklarációja egy másik típus neve, akkor a definiált típus azonos a megnevezett típussal. A típusdeklarációk a TYPE kulcsszóval kezdődnek, és a következő deklarációs listáig tartanak. A modulok fejlécében tetszőleges számú típusdeklarációs lista helyezhető el.

Formálisan:

$ típusdeklaráció = azonosító „=" típus.

$ típus = egyszerűtípus | tömbtípus | rekordtípus | halmaztípus | mutatótípus | eljárástípus.

$ egyszerűtípus = qualident | felsorolás | intervallumtípus

Példák: m2-02.lst fájl 1.lista

Két változó akkor azonos típusú, ha a megadott típusazonosító megegyezik, vagy ha a változók egy azonosító listában szerepelnek.

Példák: m2-02.lst fájl 2.lista

Számok

Az alaptípusok közül először nézzük a Modulában használatos numerikus típusokat!

Az INTEGER típus

Ez a típus az egész számokat jelöli. A negatív egész számok előtt „-" jel áll. Az egész számok oktális, decimális és hexadecimális formában adhatók meg. Azonban a számítógépek belső számábrázolása az egész számok egy véges halmazára korlátozza az INTEGER típus értéktartományát, amely általában -2^N-1 és 2^N-1 -1 közé esik, ahol N a számítógép szóhossza bitekben (általában 8, 16 vagy 32 bit). Ha olyan egész számokat használunk, amelyek kívül esnek ezen az értéktartományon, illetve ha egy aritmetikai művelet eredménye esik ezen kívül, akkor a számítógépben túlcsordulással „figyelmeztet". Ilyenkor általában (jobb esetben) a program futása megszakad. (Azért mondjuk, hogy jobb esetben, mert ha a számítógép csendben „lenyeli" a túlcsordulást, akkor hosszú idő telhet el, amíg a rejtelmesen viselkedő vagy furcsa eredményeket produkáló program hibáját megtaláljuk. A PC sajnos a „lenyelős" fajtához tartozik [lásd: m2-02.lst fájl 3.lista]). A PC-s implementációk az INTEGER típust 16 biten (ahol egy gépi szó két bájt) kettes komplemens kóddal ábrázolják, így

MinInt = MIN(INTEGER) = -32 768 és MaxInt=MAX(INTEGER)=32 767.

Az INTEGER típus megfelelője Pascalban az integer, C-ben pedig az int típus.

A CARDINAL típus

A gyakorlatban sok probléma megoldható pozitív egész számok használatával. Így a Pascalhoz képest a Modula-2 alaptípusai a CARDINAL típussal bővültek. A CARDINAL típus pozitív egész számok (és a 0) ábrázolására szolgál. A CARDINAL típus használatával a programozó a változókat explicit módon csak pozitív értékeket felvevőknek deklarálja. Előny, hogy az értékek halmaza nagyobb, mint pozitív INTEGER típusú számok esetén (a maximális érték általában kétszeres, azaz 2^N -1). Az IBM PC-s implementációk a CARDINAL típust 16 biten (ahol egy gépi szó két bájt) ábrázolják, így

MinCard=MIN(CARDINAL)=0 és

MaxCard = MAX(CARDINAL) = 65 535.

A CARDINAL típusú számok megadási módja az INTEGER típushoz hasonlóan lehet oktális, decimális és hexadecimális. A CARDINAL típusnak

Pascalban nincs megfelelője, C-beli megfelelője az unsigned int.

A LONGINT típus

A gyakorlatban igény van az egész számok halmaza egy nagyobb intervallumának ábrázolására is. Ugyanis az INTEGER típus még az IBM PC csekély memóriaméretének a reprezentálására sem elég. Ezen a gondon segít a LONGINT változó, amely az egész számok egy nagyobb halmazát deklarálja, amely általában -2^2*N-1 és 2^2*N-1 -1 közé esik. Így a PC-s implementációkban

MinLongInt = MIN(LONGINT) = —2 147 483 648 és

MaxLongInt = MAX(LONGINT) = 2 147 483 647.

Sajnos ez az összeg sem lenne elegendő Magyarország adósságállományának nyilvántartására, de a legtöbb alkalmazáshoz kielégítő. Ha valaki mégis segíteni szeretne az éppen aktuális pénzügyminiszternek az adósságállomány fillérekig pontos kezelésében, akkor vagy válogathat a piacon kapható többféle BCD könyvtári modul között, vagy pedig kivárja, amíg a Modula-2 sorozatban kifejlesztünk ilyet.) A LONGINT típusnak Pascalban nincs megfelelője, C-beli megfelelője a long típus.

A LONGCARD típus

Az értéktartomány kiterjesztése a természetes számokra is vonatkozik. Bár a Wirth-féle Modula-2-definícióban nem szerepel a LONGCARD típus, mégis a Modula-2-implementációk nagy része tartalmazza ezt a bővítést. Akárcsak a LONGINT típust, a LONGCARD típust is általában két gépi szóban ábrázolják. Így a PC-s implementációkban

MinLongCard = MIN(LONGCARD) = 0 és

MaxLongCard = MAX(LONGCARD) = 4294967295.

A LONGCARD típusnak Pascalban nincs megfelelője, C-beli megfelelője az unsigned long típus.

Az IBM PC-s implementációk az Intel 80x86 típusú gépek sajátosságait kihasználva bevezették a 8 biten ábrázolt SHORTINT és SHORTCARD típusokat is.

Az INTEGER és CARDINAL típusokat együttesen egészszámtípusnak nevezzük.

Példák: EX02-01.MOD, EX02-02.MOD fájlokban.

Az egész számokra értelmezett műveletek:

Összeadás (+) pl: 5+4, 13 + 2

Kivonás (-) pl: 16-11, 9 — 12

Szorzás (*) pl: 7*5, 4 * 12

Egész osztás (DIV) pl: 15 DIV 3,17 DIV 5

Maradékképzés (MOD) pl: 15 MOD 3,17 MOD 5

(Megjegyzés: a DIV és MOD operátorokat az operandusoktól szóközzel kell elválasztani. Ez a jelölés a többi operátornál nem kötelező, de a jobb olvashatóság miatt célszerű.)

Az egészszám típusokra alkalmazható standard eljárások:

Abszolút érték: ABS (x)

Páratlan/páros: ODD (x)

Sorszám: ORD (x)

Dekrementálás: DEC (x)

n-szeri dekrementálás: DEC (x,n)

Inkrementálás: INC (x)

n-szeri inkrementálás: INC (x,n)

A valós számok

A Modula-2-ben a valós számokat a REAL típus reprezentálja. A valós számok a törtekkel való számolást és az egészszámtípusnál nagyobb számokkal való számolást teszik lehetővé. Ez azonban csökkenti a számolás pontosságát. Ugyanis a valós számokat lebegőpontos számként ábrázolják m*2^e alakban, ahol m a mantissza, e az exponens értéket tárolja véges számú biten. A véges szóhosszúságú tárolás miatt a valós számokkal végzett műveletek eredményei az esetleges kerekítések miatt pontatlanok. A REAL típus ábrázolása gépfüggő. Már létezik ugyan IEEE szabvány a lebegőpontos számok ábrázolására, használata azonban nem terjedt el. Akárcsak az egészszám típusoknál, sok Modula-2-implementációban a valós számok esetében lehetőség van LONGREAL típussal nagyobb értéktartományú és pontosságú számok használatára. (Ez a bővítés nem általános, mivel sok gépen a REAL típus eleve a maximális értékkészletet és pontosságot jelenti.)

A valós számok megadhatók tizedes szám alakjában vagy (tízes alapú) exponenciális alakban. A tizedesszámalakban a tizedes pont és az egész rész megadása kötelező. Exponenciális alakban „E" betű jelöli a tízes hatványt, az „E" mögött álló előjeles egész szám a kitevőt.

A TopSpeed Modula-2-ben a két tartomány

REAL: +/-1.2E-38 — 3.4E+38 a pontosság 6 számjegy

LONGREAL: +/-2.3E-308 — 1.7E+308 a pontosság 15 számjegy

A REAL típus megfelelője Pascalban a real, C-ben a float típus. A LONGREAL típusnak Pascalban nincs megfelelője, C-ben a megfelelő típus a double.

Példák: az EX02-03.MOD fájlban

A valós számokra értelmezett műveletek:

Összeadás (+) pl: 15.3 + 7.5, -12.9E3 + 12.4E2

Kivonás (-) pl: 12.4 — 3.4, -23.987E-2 — 2.5E-1

Szorzás (*) pl: 3.3 * 1.2, -3.7E1 * 12.0E2

Osztás (/) pl: 12.4 / 3.3, -2.3 / 2.4E-9

A valós típusokra alkalmazható standard eljárás: abszolút érték: ABS (x)

Minden Modula-2-implementációhoz tartozik egy MathLib0 nevű könyvtári modul, amelyből az alapvető matematikai függvények importálhatók. A modul nevében az alapfunkciókra a 0 utal, és általában a MathLib nevű modul tartalmazza az egyéb függvényeket. (A lemezmellékleten megtalálható a Wirth-féle Modula-2 MathLib0 könyvtár definíciós modulja.)

A különböző egész, természetes és valós számok együttes elnevezése numerikus típus.

Típuskonfliktus és típuskonverzió

A Modula nem teszi lehetővé kevert kifejezések használatát, vagyis eltérő típusú operandusokat kifejezésekben. Ennek egyik oka a Modula szigorú típusellenőrzése, a másik ok pedig az, hogy az eltérő típusokra vonatkozó gépi kódú utasítások eltérőek. Az alaptípusok szigorú elkülönítése (különösen például az INTEGER és a CARDINAL esetében) túlzottnak tűnhet, elsősorban C programozók számára. Ha azonban nem az egy-két soros programkezdeményekre gondolunk, talán már nem is olyan felesleges luxus. Az, hogy a számítógépben eltérő módon reprezentált valós és egész számokkal való kevert műveletek nem megengedettek, nem hökkent meg senkit. De miért nem megengedett a 16-16 biten tárolt egész típusok keverése? A kérdésre egyszerű a válasz. Egy adott probléma megoldásakor a deklarált változók típusát valamilyen tervezési meggondolás alapján választjuk ki. Ha egy CARDINAL-nak deklarált változót INTEGER típusúakkal együtt kell használni, akkor a típusválasztás nem volt megfelelő, és célszerű ezt módosítani. Ez az eljárás minden típusra érvényes. A döntés, hogy egy változót INTEGER vagy REAL típusúnak deklarálunk, nem a lefoglalt bájtszámon múlik, hanem azon, hogy az adott változón a probléma

megoldásakor milyen műveleteket végzünk el. (Például logaritmus-számításhoz ne használjunk CHAR változót!) A típuskonfliktusok az esetek nagy többségében megfelelő tervezéssel kiküszöbölhetők.

Természetesen, ha ezt a szigorú típusellenőrzést nem lehetne feloldani, akkor senki sem programozna Modulában. A megoldás egyszerű. A Modula úgynevezett típusátalakító függvényeivel tetszőleges típus tetszőleges típusra alakítható át. Ezek a függvények persze csak kvázi függvények, mivel valóságos feldolgozást nem hajtanak végre, csak a fordító szigorú típusellenőrzését kerülik meg. A függvényeket egyszerűen típusazonosítóval deklaráljuk, az átalakítandó objektum ennek a kvázifüggvénynek az argumentuma. A típusátalakítás szabályai tetszőleges típusra érvényesek. Ezek a kvázi-függvények a változókat nem konvertálják a céltípusra, csak úgy értelmezik. (A C-fordítók az átalakítást a háttérben végzik.) A típusátalakítás minden esetben gondot okoz a program működésének elemzésekor. Azokban az esetekben, amikor a két típust eltérő számú bájton ábrázolja a számítógép, a bájtok „elvesztése" (átalakítás hosszabbról rövidebbre), illetve a nem definiált bájtok (átalakítás rövidebbről hosszabbra) okoznak gondot. Azonos hosszúságú típusok esetén sem mindig egyértelmű az eredmény. A típusátalakításból származó problémákat az m2-02.lst fájl 4.listája szemlélteti.

A fenti hibákat küszöbölik ki az úgynevezett típuskonverziós függvények. Ezek valódi függvények, az eltérő reprezentálásból adódó hibákat megszüntetik. A standard konverziós függvényeken kívül a legtöbb könyvtár saját eljárásokat is szolgáltat az ellenőrzött típuskonverziók elvégzésére.

A standard konverziós függvények:
CARDINAL-REAL: FLOAT (cardVar)
REAL-CARDINAL: TRUNC (realVar)
TYPE1-TYPE2 : VAL (TYPE2, type1Var)
Példák: M2-02.LST fájl 4.lista és EX02-04.MOD fájl.

A különösen C-programozók számára körülményesnek tűnő típuskonverzió a programtervezést és nem az „essünk neki, holnap van a határidő" típusú kódolást támogatja. Előfordulnak olyan esetek, amikor optimális tervezés ellenére is szükséges eltérő típusokon műveleteket végezni. C nyelvben ilyenkor a fordító csendben átsiklik a típus-inkonzisztencián, mivel C-ben a típusazonosság nem kritérium. A fordító vagy a linker jobb esetben persze figyelmezteti a programozót a hibára, de a C filozófiája alapján a döntés teljes egészében a programozó kezében van, és ez így helyes. A programozó pontosan tudja, mit akar, már ami az éppen aktuális időpontot illeti. Persze hogy egy új verzióhoz szükséges módosításkor vagy rejtélyes hibák keresésekor emlékszik-e arra, eredetileg mit miért tett, az már kétséges.

Az, hogy a programozó nem mindig áll a helyzet magaslatán, abból adódik, hogy a C nyelv nem követeli meg az explicit típuskonverziót. Mivel a programozók általában lusták, amit nem muszáj megtenniük, azt nem teszik meg. A Modula, akárcsak a C, a programozó kezébe helyezi a döntést. Mivel más kárán tanul az okos, ezért nem söpri a problémát a szőnyeg alá. A Modula-fordító tehát kevert kifejezés esetén megáll, jelzi a programozónak a potenciális hiba lehetőségét. Ezután a programozó dönti el, hogy mit tegyen. Lehetőséget kap a koncepció újragondolására. Erre az egyik lehetséges megoldás a gondot okozó kifejezés megszüntetése, de ez nem mindig megfelelő. A másik megoldás az explicit típuskonverzió vagy típusátalakítás használata. Ez eddig nem nagyon különbözik a C nyelvi megoldástól. Az eltérés csak ezután jön.

Nézzük, hogy mi történik, amikor egy programban egy nappal a leadási határidő előtt, az utolsó módosítás hatására kevert típusú kifejezést hoz létre a programozó! A C-programozó általában egy vállrándítással veszi tudomásul a figyelmeztetéseket, hiszen a program kész, másnap leadható, és jöhet a „mani". Egyébként is a tesztprogram „csont nélkül" lefutott. A szerencsétlen Modula-programozó viszont elkezd izzadni, hiszen éjfél van, és a nyomorult fordító hibát talált a kódban. Modulásunk kimegy a konyhába, és miközben felteszi a kávét, azon gondolkodik, melyik közeli rokonának a temetésére hivatkozva mondja le a másnapra tervezett leadási ceremóniát. A fekete nedű szürcsölése közben átfutja a szerződés kötbérről szóló részét. Aztán egy nagy sóhajtással nekifog a hiba elhárításának. Természetesen elvégzi a megfelelő típuskonverziót vagy típusátalakítást. Ha már úgyis megakasztották a munkában, végiggondolja a konverzióból származó lehetséges hibákat, elvégzi az elhárításukhoz szükséges módosításokat úgy, hogy a kifejezés mindig megfelelő eredményt produkáljon. (És míg a C programban semmi sem utal a típusmódosításra, hiszen de facto meg sem történt, addig a Modula program forráskódjában ott „éktelenkedik", szinte kiabál a típusmódosításra utaló nagybetűs típusazonosító, amit egy esetleges későbbi módosításkor nem lehet elkerülni.) Egy héttel később Modulásunk a Hawaii-szigeteken képeslapot fogalmaz barátjának, aki szintén programozó, és valamilyen új program bétatesztelése miatt nem tudott vele jönni. Ugyanebben az időben a C-programozó, aki persze nem más, mint a fent említett barát, immár századszor nézi át újra és újra a forráskódot, és halkan mormolja maga elé: „nem értem, a tesztprogram hiba nélkül lefutott". Na igen, de a felhasználót semmi sem akadályozta meg abban, hogy olyan értékeket adjon a változóknak, amelyek hatására a típusátalakító műveletek nagy része nem az elvárt módon működött.

Ismerős a szituáció? Talán igen, talán nem. Nem állítom, és nem állíthatom, hogy egy adott nyelvet használó programozó ilyen vagy olyan minőségű munkát végez, vagy ilyen vagy olyan hatásfokkal dolgozik. Már korábban leírtam és hiszem: a programozó munkáját befolyásolja, hogy munkája során milyen programozási nyelvet használ. Ha a választás szerencsés, akkor megkönnyítheti a problémamegoldást és megnövelheti a hatékonyságot. Ez nemcsak a típuskonverziók esetén van így, arra példa lehet móodjuk egy C-ben írt adatbázis-kezelő (nevezzük Flitter Nyár 87-nek). Beharangozott új verziója (nevezzük Flitter 5.0-nak) kétéves késés után mindössze annyiban különbözik az előző verziótól, hogy a felduzzadt fejlesztési költségek miatt a duplájába kerül, és tele van hibával. Vagy vegyünk például egy assembly-nyelven megírt táblázatkezelőt (nevezzük Kaktusz 1-2-3 2.0-nak), amelynek nagy csinnadrattával bejelentett új verziója (Kaktusz 1-2-3 3.0) egy év késés után, a beígért bővítéseknek csak egy töredékét tartalmazza.

Az Alaplap júliusi lemezmellékletén elhelyezett példaprogramok numerikus típusok használatát mutatják be. Ehhez kapcsolódik a utility program is, amely egy számológép-kezdemény. A program csak az alapvető számítások elvégzését támogatja, de bárki tetszőlegesen továbbfejlesztheti saját céljaira.

Villányi László

1991. július 1-jétől
a **Cédrus** Informatikai Rt.
új címe: 1251 Budapest
XI., Karolina út 17.
Telefon: 166-2111
Fax: 185-2221

Standard Clipper-osztályok

Cikksorozatunk előző fejezetét a változókezelés tanulmányozásával fejeztük be, a harmadik részt egy speciális változótípusnak: az objektumnak szenteljük. Tesszük ezt egyrészt azért, mert a Clipper-felhasználók nagy része még nem sok helyen találkozhatott objektumokkal és objektumosztályokkal, másrészt pedig azért, mert a nyelvben ez a legnagyobb előrelépés az előző verziókhoz képest.

A Clipper 5.0 az alkotók bevallása szerint sem objektumorientált programozási nyelv, de kétségtelen, hogy az első lépéseket már megtette ebbe az irányba. Megjelent benne egy új változótípus: az objektum. Sajnos egyelőre nincs lehetőség saját objektumosztályok definiálására, csak azt a négy osztályt használhatjuk, amit a Nantucket standard osztálynak nevezett el, és beépített a nyelvbe. Ez a négy osztály a következő: errorclass standard hibaosztály, getclass standard getosztály, tbcolumnclass standard adatmegjelenítési osztály, tbrowseclass standard adatáttekintési osztály.

Bár ez a négy osztály nem valami sok, de ha nem a Nantucket tette volna meg, akkor mi is ezt a négyet definiálnánk először. A standard osztályok nyelvi megvalósítása elég jónak mondható, az objektumokkal végezhető műveletek száma megfelelően nagy. Jó volna — ha már saját osztályok definiálását nem tették lehetővé —, ha legalább a standard osztályokhoz készíthetnénk saját funkciókat vagy deklarálhatnánk saját változókat. Az elszalasztott lehetőségeket csak igen csekély mértékben kárpótolja a mind a négy osztályban használható cargo nevű változó. Ám ennek használata is rejteget bizonyos előnyöket. Igénybe vehetjük különféle üzenetek vagy kódok küldésére az objektum függvényei számára. Túlzott elkeseredésre azért sincs okunk, mert a legtöbb (és minden általánosan használt) funkciót megvalósították a standard osztályokban. Ha a lehetőségekhez képest jól használjuk ki őket, akkor már elégedettek lehetünk magunkkal, és meg merem kockáztatni, hogy a felhasználók is elégedett(ebb)ek lehetnek velünk.

Mivel az objektum- és az osztály-fogalmakat még nem minden Clipper-felhasználó ismeri, nem árt röviden kifejteni, hogy mit is jelentenek. A száraz, tömör és nehezen emészthető definíciók helyett inkább próbáljuk meg nagyító alá venni a Clipper előre definiált osztályait. A legegyszerűbb és talán a leggyakrabban használt objektumosztály az errorclass. Csak 13 exportált változója van, és mindössze egy ritkán használt osztályszintű függvény tartozik hozzá. Működésének lényege a következő: ha a rendszer akármilyen runtim-hibát észlel, új hibaobjektumot generál, annak összes exportált változóját kitölti az észlelt hibát leíró értékekkel, majd az objektumot mint változót (igazából az objektum címét) argumentumként átadja annak a kódblokknak, amelyet az ERRORBLOCK( ) függvényben meghatároztunk.

Ez a kódblokk aztán általában (az alapértelmezett kódblokk megszakítja a programot) meghív egy hibakezelő függvényt, amelynek továbbadja az objektumot (az objektum címét). A hibakezelő a hibaobjektum változóinak tartalmából kihámozza a hiba okát és jó esetben azt is, hogy milyen akciót lenne még érdemes elindítani.

Másodiknak a GET osztályt vizsgáljuk meg, amely kicsit bonyolultabb, mint az előző. Ebben az osztályban rendelkezésünkre áll 19 exportált változó, 9+7+5+2=23 exportált eljárás és egy osztályszintű függvény. Az objektumosztályt az előző változatok GET—READ típusú adatbekérési mechanizmusa helyett vezették be. Az előző változatokkal való kompatibilitás megőrzéséhez az std.ch include-állományban (323.sor) definiálva van a GET és a READ átalakítása az új szisztémának megfelelő függvényhívásokra. Az új szisztéma szerint GET osztályú objektumokat, minden GET mezőhöz egy objektumot kell létrehozni és az exportált változókba nekünk kell beírni a kívánt paramétereket (oszlopszám, sorszám, az adott GET listaelemre vonatkozó színmeghatározások, picture-kódok és -maszkok stb). Az összeállított GET objektumokat egy tömbbe kell foglalni, majd ezt a tömböt átadni a beolvasást végző READMODAL( ) függvénynek.

Ez a megoldás több, jól használható lehetőséget rejt magában az előző változathoz képest. Például a GET-eket egymásba lehet ágyazni, mert a READMODAL( ) csak azokat a GET objektumokat dolgozza fel, amelyek az argumentumtömbben benne vannak. Az egy READMODAL( ) függvénnyel behívott változók beolvasása között végrehajtathatjuk saját függvényeinket (mértékegység–átszámítás, adathelyesség-vizsgálat, esetleg megerősítési kérelem a felhasználó számára vagy egy felfedezett hiba speciális kezelése), sőt: megadhatunk egy függvényt, amelyet az adott GET mezőbe való belépéskor, és egy másikat, amit kilépéskor kell végrehajtani.

A maradék két objektumosztály, a TBROWSE és TBCOLUMN osztályok szorosan összefüggenek egymással. Leírásuktól most eltekintünk, mivel a rendszer részét képező TBDEMO.PRG közel hat kilobájton mutatja be használatukat, és egyszerű példát találunk benne egy GET objektum kezelésére is. Egyébként az objektum- és az osztályfogalmaknak, valamint a Clipper 5.0 standard osztályai használatának a megismerésében segítségünkre lehetnek még a következő (a rendszer részét alkotó) programok is: BROWSE és COLUMN: array.prg, dbuedit.prg, GET: getsys.prg, valedit.prg, ERROR: errorsys.prg, frm????.prg, lbl????.prg.

Fridl György

Programozás és a józan paraszti ész?

Nem könnyű szerkesztői bevezetőt írni az alábbi írás elé. (Aki elolvassa, rögtön megérti, hogy miért.) Egyrészt örülünk, hogy akaratlanul is sikerült a programozástechnikával kapcsolatban egy igen érdekes szakmai állásfoglalás kifejtését „kiprovokálni", másrészt nem szeretnénk, ha az Alaplapban közölt írások vitastílusa — tárgyukhoz nagyon méltatlanul — személyeskedő, sértegető jelleget öltene. (Még akkor sem, ha a fő vitapartner tulajdonképpen egy külföldi lap sehol meg nem nevezett szerzője.) A cikkben előforduló ilyen stiláris fordulatokat ezért többnyire kigyomláltuk, de csakis azokat! A téma önmagában is elég érdekes ahhoz, hogy higgadt, tárgyilagos eszmecserére serkentsen olyanokat, akik szerzőnkhöz hasonló szeretettel kezelik hivatásuk eszköztárát. Szívesen helyt adunk tehát a további véleményeknek is.

Az Alaplap áprilisi számában „A problémától a programig" címmel, a Programozástechnika rovatban jelent meg egy írás a Toolbox magazin cikke alapján. Sajnálatos félreértés miatt a cikk bevezetőjében az állt, hogy a cikk „problémafelvetését tekintve a Modula-sorozat megállapításaival is vitatkozik". Bár jól tudom, hogy a fenti cikk nem az Alaplap Modula-sorozatára reagál, mégis úgy éreztem, fel kell vennem a kesztyűt (vagy talán inkább csak a feladott labdát kell leütnöm).

Az idézett mondat már csak azért is felkeltette a figyelmemet, mert a Modula-sorozatban igyekeztem kerülni mindenféle általánosító megállapítást, és persze kíváncsi voltam azokra a bizonyos vitatkozó problémafelvetésekre. Nézzük hát a cikket! Itt van mindjárt a bevezető: „15 évvel ezelőtt a programozást a Fortran és a hozzá hasonló nyelvek, a lyukkártyás technika jelentette". Gyors fejszámolás: 1991 – 15 = 1976. Hm. A Fortran és a hozzá hasonló nyelvek? Mármint COBOL (1961), ALGOL (1960), LISP (1960), BASIC (1964) vagy a többiek, mint APL, PL/1, SIMULA, Forth, PASCAL, SMALLTALK, és már készült a C és a Modula-2 megjelenése is. Vajon milyen új, azóta történt jelentős nyelvi fejlődésre gondolhatott a cikk írója ezeken a nyelveken kívül?

Hm, másodszor: „lyukkártyás technika". Mármint a CP/M mint a 8 bites gépek elfogadott szabványa vagy a 6502-es processzorra épült Apple-gépek — amelyek már régen nem abban a legendás garázsban készültek —, hogy a virágzásnak indult hobbi- és mikroszámítógépek világából csak néhány példát említsek, vagy — mondjuk — a hódító útján már öt éve elindult UNIX operációs rendszer. (Higgyék el, kedves olvasóim, hogy nem a lyukkártyás technika nagyszerű megoldásai azok, amelyek ezt az operációs rendszert olyan elterjedtté tették szerte a világon az azóta eltelt évek alatt.) De a nagygépes hardveroldalon is bőven lehetne sorolni a különféle géptípusok és gyártók neveit, amelyek bizony már rég elfelejtették a lyukkártyaolvasót mint elsődleges adatbeviteli eszközt a felhasználóval való kommunikációban.

Ezek után az az állítás, hogy „a feladatok megoldásakor nem volt szükség az akkori legújabb szoftvertechnológia ismeretére", nemcsak téves, hanem egyenesen sértő is. Meg aztán, hogy a nyolcvanas évek sem hoztak minőségi változást? Azok a nyolcvanas évek, amelyek mindenki asztalára egy korábban el sem képzelhető számítógép-kapacitást helyeztek, lehetővé téve a számítógép használatát az élet szinte minden területén, és így a korábbinál sokkal többen ismerkedhettek meg a programozás világával is!

Természetesen az, hogy valaki számítógéppel rendelkezik, még távolról sem jelenti azt, hogy programozóvá válik. Sőt: azért, mert valaki programokat ír, még nem biztos, hogy programozó. De az biztos, hogy a sokféle új felhasználói igény és a személyi számítógépek kapacitása olyan feladatot rótt a programozókra, amihez fel kellett nőni. Hogy csak egy iciri-piciri példát említsek: gondolom, a cikk szerzője sem soreditor segítségével készítette el írását. Az viszont, hogy a programozási nyelvek sorába egy gépi kódú fordító (assembler) is bekerült, lehet tévedés is. Gondolhatják, hogy ilyen bevezető után milyen izgalommal vártam a folytatást.

A cikkben a nyolcvanas évek végére datált programozástechnikai fogalmakról csak annyit, hogy azok már a hatvanas évek végétől, az úgynevezett „nagy szoftverválság" időpontjától ismertek, és — akár hiszi valaki, akár nem — a professzionális programozók használták is őket. A józan paraszti észre való hivatkozást egyszerűen nonszensznek tartom. Ebből sajna egy szó sem igaz. Egy programozó munkája során nem a józan paraszti észre, hanem szaktudására, a gyakorlatban megszerzett rutinjára és az adott probléma megoldásával kapcsolatos speciális ismereteire hagyatkozhat. A jó értelemben vett intuíciót és kreatív készséget pedig nem sorolnám a mezőgazdasági szakismeretek körébe. Vagyis tagadom, hogy a programozói munka józan paraszti ésszel elvégezhető. Már csak arra lennék kíváncsi, hol van a „programozók nagy része", hol vannak azok, akik erre esküsznek.

Az objektumorientált programozás sem kerülte el a szerző figyelmét, de sok jót erről a témáról sem tudott mondani. Van, amiben természetesen egyetértek: az, hogy valaki ilyen vagy olyan programozási nyelvet használ, önmagában még nem jelent semmit. Megnyugtathatok mindenkit, azért, mert mondjuk valaki Basic-ben ír programokat

akár a legcsekélyebb programozási alapismeret nélkül, számomra nem jelent mást, mint egy vállalkozó szellemű és az új dolgokra fogékony személyt. Ha például az illető néhány sikeres kísérlet után lelkesen bizonygatja a Basic nyelv szépségeit és előnyeit, még nem tartom eszelősnek. Magam is jól emlékszem a jó öreg ABC 80-on megírt jelanalízist végző programjaimra, amelyeket természetesen Basic-ben írtam, lévén az egyetlen hozzáférhető nyelv. És mivel semmi problémám nem volt az A/D átalakító kártya programozásával, én is nagyon elégedett voltam. Ha az illető egy-két hónap elteltével előállna azzal az ötlettel, hogy egy real-time többtaszkos operációs rendszer megírását forgatja a fejében, alighanem elejtenék néhány célzást a Basic korlátaival kapcsolatban, de nem óhajtanám ezért a dilihászba csukatni. Ha erre az ötletére céget akarna alapítani, az egyetlen, amit tehetek, hogy anyagilag nem támogatom a vállalkozását. Mindezt csak azért tartom fontosnak elmondani, mert — és ebben egyetértünk a szerzővel — a „nyelvi villongások" abszolút értelmetlenek, és nem vezetnek sehová. Mindenkinek szíve joga programozási nyelvet és metodikát választani, az elkészült program minősége azonban nem a józan paraszti észtől függ. Ezen túl persze más valaki munkájának, programozási módszerének a minősítése, hacsak az nem felkérésre történik, nem a mi feladatunk, hiszen, mint tudjuk, fogadatlan prókátornak... A cikk első részének taglalását ezzel be is fejezem, mivel — több megállapítás nem lévén — további félrevezető információkat már nem tartalmaz.

A cikk elejének fanyalgásai után jött a „bizonyítás". Bár az, hogy végül is mi a cikk írójának a véleménye, és valójában mit akar bizonyítani, nem derült ki. A példaprogrammal minden rendben van, csak azt nem látom be, hogyan lehet egy pár soros programocska segítségével a cikk elején említett programozástechnikai fogalmakat pro vagy kontra demonstrálni. Amennyiben a programozó és a programozás fogalmát csak ilyen „nagy ívű" feladatokkal kapcsolatban értelmezi, akkor ugyancsak felesleges volt ezeket megemlítenie.

A lineáris programváltozatról szóló részt mindjárt egy belső eljárás ismertetésével kezdi. És itt meg is állhatunk. További boncolgatások nélkül leszögezhetjük: lineáris programozás magas szintű nyelven nincs. Az első implicit vagy explicit függvényhívással máris átléptünk a szerző által procedurálisnak nevezett programozás területére, mert az, hogy egy adott eljárást mi írtunk vagy más valaki írt meg, nem számít különbségnek az elvek szempontjából. Továbbmegyek és azt állítom: nemcsak hogy a procedurális, de a moduláris programozásba léptünk át szinte észrevétlenül, hiszen a write eljárás megfelelői sok programozási nyelvben külön modulban találhatók. Persze, az már rajtunk múlik, hogy az általunk írt kódrészlet lineáris, procedurális vagy moduláris lesz-e. Ha valaki egy rövid programot ír, nem valószínű a modulokra való bontás, és nem kell feltétlenül minden kis logikai egységből eljárást kreálni (a hangsúly a rövidségen van). De ha valaki programozó, akkor nem a rövid programok írása dominál munkájában. Ezért egy komplett rendszerben a logikailag elkülöníthető egységeket eljárásokba, a logikailag egy csoportba sorolható objektumokat modulokba foglalja. Ha valaki ezenfelül professzionális programozó, akkor az eljárásokat és a modulokat általános célúaknak definiálja, ezzel is biztosítva a későbbi felhasználhatóságot. Ha valaki a programozásból él, akkor saját modulkönyvtárát igyekszik mások számára (természetesen megfelelő ellenszolgáltatásért) hozzáférhetővé tenni, de mivel a világegyenlet megoldó képletét és más fontos problémamegoldást nincs ideje lekódolni, ezért nagy hasznát veszi a mások által már megírt könyvtáraknak is. A programozók világa rendeltetésszerűen moduláris (de nem Moduláris!). Azt hiszem, az elméleti részt ezzel le is zárhatjuk.

Lássuk csak az implementációt! Azt hiszem, ez az a rész, ami a szerkesztőt megtéveszthette és a bevezető záró-mondatának megírására késztethette. Természetesen én is izgatottan vártam a cikkben a Modula színrelépését. Sajnos nagyot kellett csalódnom. Miután a szerző megemlítette a Modulát, azonnal félretette, és helyette előkerítette a Turbo Pascalt. Maga a szerző is beismeri, hogy ez a Pascal-implementáció csak „valami hasonlót nyújt", mint a Modula.

Véleményem szerint az, hogy a 4.0-s verziót megszületésekor nem Turbo-Modulának hívták, valójában csak az addigra már kialakult piaci helyzetnek volt köszönhető. Persze ennek ellenére elég sok mindent megpróbáltak átvenni a Modulából. Az eredmény eléggé kétes, de amint a Pascal-bővítésekről írtam, egy jó programozó még egyáltalán nem biztos, hogy jó nyelvtervező is. Azzal, hogy kétes állításai igazolására a cikkíró nem egy moduláris nyelvet választott, szerintem nagyot vétett az objektív vizsgálati módszer követelményeivel szemben. Sebaj, a békát lenyelem, ha nehezemre is esik. Így lőn a Lib UNIT. És lám, következett az eredménykiértékelés is, amelynek lényege, hogy „lássuk a medvét", azaz a keletkezett EXE fájlok méretét. Az eredmény:

„Lineáris" 4976 bájt.
„Procedurális" 5063 bájt.
„Moduláris" 5209 bájt.

Majd jön a verdikt: „Ha így nézzük, egyértelműen a ... lineáris program lenne a nyerő". De kérdem én, miért így nézzük? Ma, a 16 megabájtos 286-osok és a 4 gigabájt virtuális memóriát kezelő 386-osok világában — hogy a Motorolákról és a RISC processzorokról ne is beszéljünk — az a 200 bájt ugyan mit számít? Főleg ha hozzátesszük azt is, hogy a programok bizony nem tökéletesen ugyanazt csinálják, ezért nem alkalmasak az összehasonlításra. De talán hagyjuk a Pascalt, és nézzünk meg egy valóban moduláris nyelvet, a Modula-2-t. A programok Modula-változatai a lasc, pasc és masc (illetve mlib.def és mlib.imp) fájlokban találhatók a mágneslemez mellékleten. Ha jól megnézzük, rájöhetünk, hogy a Modulával jóformán ha akarnánk, se tudnánk „lineáris" programokat írni. A Modulában szinte minden eljárás valamilyen könyvtármodulból importálható. Nincs szükség „Error" eljárás írására, hiszen már készen „kapható", akárcsak a többi eljárás. Ezért a Modula-kódok között alig akad különbség. A Modula maga a moduláris programozás.

És az eredmény? Azt hiszem, önmagáért beszél:

„Lineáris" 2980 bájt.
„Procedurális" 2988 bájt.
„Moduláris" 2970 bájt.

Azaz tökéletesen megegyező. A magam részéről természetesen a lineáris megoldást választom, hiszen az adott problémát ez a módszer oldja meg a legegyszerűbb módon. Ha nem kell, ne csicsázzuk feleslegesen a dolgokat. Ezután jön az elengedhetetlen „persze assemblyban mindez sokkal rövidebb". Hát amennyiben a kódméretre gondolunk, ez tényleg így van. A kódolás sokkal hosszabb, a méretre vonatkozó 200 bájtos saccolás (ráadásul konkrét program nélkül) igencsak sántít, és ne feledjük el, hogy az exe programjainkat tároló fájlok 512 bájtnyi fejléccel rendelkeznek a relokálhatóság érdekében. Persze még mindig ott van a szükséges rossz: a mintegy 1000 bájt (hűha!!) hosszú iniciáló rész, de ez egyrészt 100 kilobájtos programok esetén sem lesz nagyobb, másrészt jó azt tudni, hogy az

ember ballépéseire figyelnek. Persze szép dolog is egy ilyen programot assemblyban írni, hiszen minden felesleges dolgot elhagyhatunk, és még abszolút lineáris programozással a futásidőt is csökkenthetjük, ami ugye nyomtatás esetén rendkívül fontos dolog. És hát az égvilágon semmi másra nem alkalmas az a munka, amit elvégeztünk.

A tanulság számomra az, hogy assemblyban is a modularitás az egyik legfontosabb célkitűzés, csak éppen a megoldáshoz választott eszköz más. Mondanom sem kell, hogy a magas szintű nyelven írt programot sokkal rövidebb idő alatt tudom tetszőleges másik gépre konvertálni, mint az assemblyben írtat. Egy ilyen egyszerű program esetén még csak-csak, de probálkozzunk — mondjuk — egy szövegszerkesztő forráskódjával. A kódméretről még csak annyit: örülök, hogy az Alaplap áprilisi Modula-cikkében a Wirth professzortól közölt idézet már mindent elmondott, amit az implementáció fogalmáról el lehet mondani, és bár „A problémától a programig" című cikkről az áprilisi rész megírásakor még nem is tudtam, a válasz máris benne volt ugyanabban a számban.

Villányi László

TV kontra PC

Az alábbi kis írást a PC Turbo Klub hardverszakértője jegyzi. Egymástól függetlenül többen is tanácsot kértek tőle telefonon, illetve levélben, s úgy érezzük, a válaszok a szélesebb olvasótábor érdeklődésére is számot tarthatnak.

Sokszor vetődik fel a kérdés, hogy lehetséges-e, és ha igen, akkor műszakilag hogyan oldható meg a PC családba tartozó számítógépek videokártyáinak illesztése műsorvevő televíziókészülékhez.

Az igény általában olyan esetekben merül fel, amikor a számítógépet nagyobb távolságból nézik, például kiállításon bemutatásra használják. A különlegesen nagy méretű (19"), kiváló felbontású professzionális monitorok használata erre a kivételes alkalomra nem indokolt és igencsak borsos áruk miatt nincs is reális lehetőség beszerzésükre.

Általában elmondható, hogy bármilyen csúcsminőségű televízióval is próbálkozunk, a számítógép illesztésekor kapható képminőség nem éri el a műsorvétel során megszokottat. Kísérletezni csak alapsávi videobemenettel is ellátott (EURO—SCART, DIN vagy RCA csatlakozójú) készülékekkel érdemes, a nagyfrekvenciás moduláció és demoduláció rontaná a minőséget, bonyolítaná az illesztést.

Mérlegelni kell azt is, hogy a tv-képcsövek hosszú ideig állóképeket megjelenítve könnyebben károsodnak, mint a monitorképcsövek, ezért elkerülhetetlen a beégésük.

Tekintsük át a tényeket és lehetőségeket!

— Az Európában működő műsorvevő televíziók sorfrekvenciája szabvány szerint 15,625 kHz. A régi készülékek hátlapján ez kézzel beállítható volt, a modern televíziók viszont egy PLL áramkör segítségével automatikusan végzik el a műveletet. A PLL áramkör a sorfrekvenciának a szabványostól néhány százalékkal való eltérését kiválóan korrigálja, nagyobb eltéréssel azonban nem tud megbirkózni.

— A PC számítógépek szabványos VGA monitorai általában 31 kHz, az EGA monitorok 21 kHz, az egyszínű Hercules monitorok pedig 18,2—18,8 kHz körüli sorfrekvenciával működnek. A videorendszerek közül a CGA az egyetlen, amelynek sorfrekvenciája a televíziók számára elfogadható lehet, 15,75 kHz. (Az EGA és VGA videoadapterek jelentős része képes CGA sorfrekvenciás üzemmódban működni.)

— A régebbi gyártású CGA kártyákon, valamint az EGA kártyák egy részén található két RCA csatlakozóaljzat, amelyek egyikén (EGA kártya esetén csak CGA üzemmódban) egyszínű (!) kompozit alapsávi videojelet kapunk. Ha megelégszünk a fekete-fehér képpel, akkor a kimenet koaxiális kábelen közvetlenül rákapcsolható a televízió megfelelő bemenetére. A másik RCA aljzat kimeneti jele NTSC rendszerű színinformációt is tartalmaz, ez azonban számunkra nem használható.

— Az elterjedtebb, csak digitális kimenettel rendelkező CGA (vagy CGA üzemmódban működő egyéb) videoadapterek kimeneti jeleiből egy speciális illesztő áramkör segítségével előállítható a PAL-rendszerű kompozit videojel, azonban ennek az áramkörnek a beszerzése nehézségekbe ütközhet.

— A digitális videokimenet illesztése RGB bemenettel (EURO—SCART-csatlakozón) rendelkező televízióhoz egyszerűbb illesztő áramkörrel is megoldható. Az áramkör különálló egységként készíthető el, saját tápegységgel.

Az áramkör feladatai:

— A CGA adapter digitális TTL szintű vízszintes és függőleges szinkronjeleiből kompozit szinkronjelet állít elő, amely polaritáshelyesen a tévékészülék kompozit videobemenetére kapcsolható.

— A CGA adapter digitális TTL szintű Red, Green és Blue, valamint Intensity jeleiből elő kell állítani a televízió EURO—SCART bemeneteire illeszkedő, 1Vpp amplitúdójú analóg Red, Green és Blue jeleket, amelyek hordozzák az intenzitás információt is.

— Biztosítania kell a tévékészülék számára a kapcsolófeszültséget, amely a videobemenetet kompozit videoüzemmódból RGB üzemmódba kapcsolja.

— A vízszintes szinkronjelet eltoltan kell a tv bemenetére kapcsolni, hogy a kép szélei is jól látszódjanak a képernyőn. (Ez egyes esetekben programból is megoldható, de nagyon bosszantó, amikor bizonyos programok futása alatt a kép máshol áll, mint egyébként.)

A fentiek ismeretében lehet tehát eldönteni, érdemes-e az egyszerű kis készüléket vásárolni és a televíziót használni, vagy célszerűbb egy nagyobb méretű monitort, esetleg videoprojektort venni vagy bérelni.

Lóth Tamás

Dr. Kovács Magda: Mikroszámítógépek alkalmazása értelmező szótár II.

Értelmező szótár 1—3.
(Budapest 1991.,
LSI Oktatóközpont Alapítvány,
1087 oldal, 699 Ft)

Sorrendben negyedikként jelent meg a hatrészesre tervezett sorozat második része, a háromkötetes *Értelmező szótár.* Készítőinek célja azonos a lapunk áprilisi számában bemutatott, készülő *Alap-szótár* tervezőiével: naprakész, minél teljesebb szakszótárt adni a szakemberek és a számítástechnika iránt érdeklődők kezébe.

A három kötetben — szinonimákkal együtt — közel húszezer címszó szerepel, a későbbi kiadások során ezt a számot még növelni kívánják. A címszavak felölelik a szakma számos területét, bár böngészés közben a *hardvertémák* túlsúlyát tapasztaltam. A legtöbb címszó után szerepel az angol elnevezés, továbbá a többnyelvű szótárak címszavaira utaló kódszám. Az egyes témakörök nem különülnek el, a címszavak ábécérendben követik egymást.

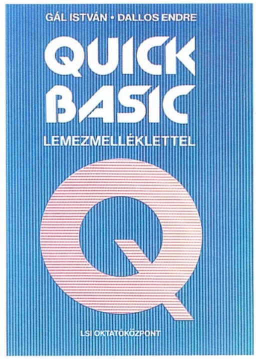
(bl)



Gál István — Dallos Endre: Quick BASIC (Lemezmelléklettel)

(Budapest 1991., LSI Oktatóközpont,
190 oldal, 369 Ft)

Érdekes könyv: szerencsésen egyesíti a szépirodalomban gyakran alkalmazott *balladai homályt* egy nem túl jól megírt felhasználói kézikönyv rosszul sikerült fordításának jellemzőivel. Alapvető információkkal marad adós, s gyakoriak az olyan kifejezések, melyekből még a témát jól ismerő is nehezen hámozza ki, mit is akarnak jelenteni. Egyetlen példa a kötet elejéről: „több értékű adattömb meghatározása". Szakértő legyen a talpán, aki ebből kisüti, hogy a „több értékű" itt *többdimenziós* tömböt jelent. Az olvasó lépten-nyomon ilyen értelmetlenségekbe botlik. Egyéni a kötet helyesírása is, különösen az ö és ő betűk használata tér el a helyesírási szótárból ismert változattól. Tárgyi tévedéseket is felfedezni véltem (például a CLEAR utasítás leírásában), de — mivel a *Quick BASIC*-nek nevezett „lényt" alig ismerem — lehet, hogy én tévedek.



A balladai homály a lemezmellékletre is jellemző. A bemutató forrásprogramokba szövegszerkesztő vagy listázóprogrammal nem lehet betekinteni, mert a Quick BASIC által kódolt (tokenizált?) formában vannak. A programok DOS alatt futtatható EXE kiterjesztésű változatait is megtalálhatjuk a lemezen, csupán egy apró információ hiányzik ezzel kapcsolatban a könyvből: elsőként a DOKUMENT programot kell lefuttatni, ennek segítségével minden további tudnivalót megkapunk a többi program futtatásához. Én — mivel a DOKUMENT látszott a legkevésbé *érdekesnek* — ezt hagytam utoljára, így a programok kezelése sikerélményt hozott számomra, *de sok fáradság árán.* Egyébként a lemezen található programok valóban érdekesek.

Egy részlet az Előszóból: „(...) merjük ajánlani a Quick BASIC-et minden olyan *magát programozónak nevező egyénnek,* aki szeretne már végre többet tudó dolgokat egyszerűbben megvalósítani." Nekem elment tőle a kedvem. Talán akad, akinek nem.

—na

BIBLIOGRÁFIA

E havi összeállításunkban az elmúlt másfél évben megjelent olyan, számítástechnikával kapcsolatos könyvek közül válogattunk, amelyekről lapunk könyvrovatában eddig még nem írtunk.

Antalóczy Zoltán (szerk.): Számítástechnika és kardiológiai alkalmazása. Budapest, 1990. Medicina, 314 oldal. Ára: 430 Ft.

Bartha Attila: Norton (Backup, Commander 3.0, Editor, Guide, Utilities 4.0, 4.50). Budapest, 1991. LSI Oktatóközpont, 168 oldal. Ára: 240 Ft.

Benkő Tiborné – Poppe András – Benkő László: Turbo C++ programozása IBM PC-n – lemezmelléklettel. Budapest, 1991. LSI Oktatóközpont, 235 oldal. Ára: 493 Ft.

Dedinszky Ferenc: Clipper 5.0. Budapest, 1991. LSI Oktatóközpont, 267 oldal. Ára: 340 Ft.

Ferenczy Imre – Gerő Judit: Quattro (kézirat). Budapest, 1991. Számalk, 104 oldal. Ára: 235 Ft.

Kollár Imre: Műszaki geometria. Budapest, 1990. Akadémiai Kiadó, 227 oldal. Ára: 198 Ft.

Lipi Gábor – Nemes Gábor – Novák István: A kínai hadseregtől az utazó ügynökig (Gráfok gépközelben). Budapest, 1990. 114 oldal. Ára: 229 Ft.

Mészáros László: Amiga DOS, Amiga Basic (Lapozgató sorozat). Budapest, 1991. Műszaki Könyvkiadó, 117 oldal. Ára: 298 Ft.

Pajzs Júlia: Számítógép és lexikográfia (Linguistica series A – Studia et dissertationes 4.). Budapest, 1990. Az MTA Nyelvtudományi Intézete, 83 oldal. Ára: 95 Ft.

Rübsam, K. M.: Merevlemezes tárolók (Alaptanfolyam 20 lépésben). Budapest, 1991. Műszaki Könyvkiadó, 119 oldal. Ára: 240 Ft.

Csábító hazai csemegetál

Mostani palettánkra szokásunktól eltérően nem hardverújdonságokat raktunk fel, hanem szoftvercsemegéket.
Ezek a szoftverek – egy kivételtől eltekintve – hazai fejlesztések eredményei. Ötletességüknél, hiánypótló szerepüknél fogva — gondolom — méltán tarthatnak számot a felhasználók táborának érdeklődésére.

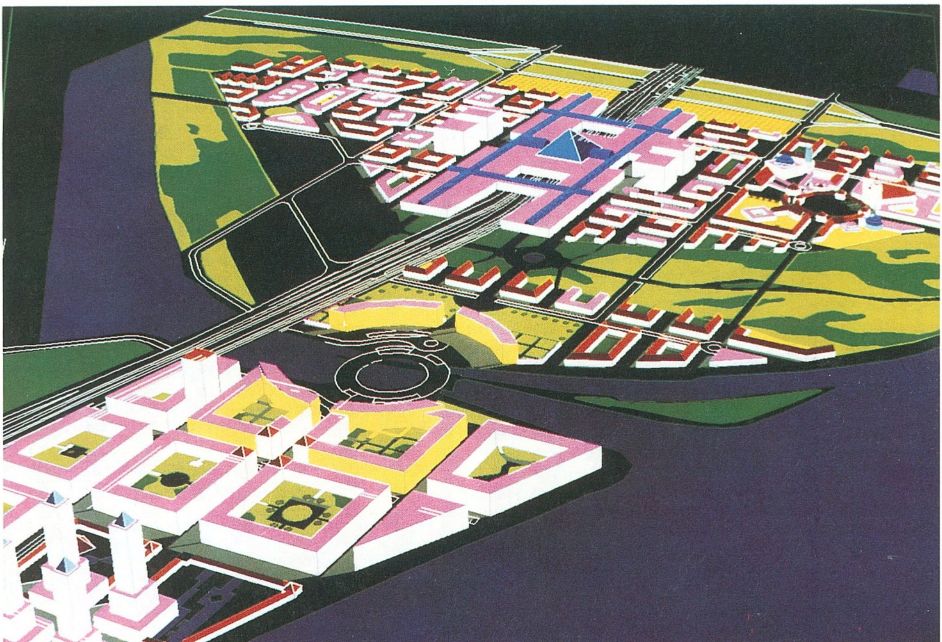
Világkiállításra fel!

A világkiállításra meghirdetett városrendezési pályázat első fordulójában részt vevő külföldi alkotók számítógép segítségével készítették el terveiket. A magyar pályázók között csak néhányan használták a számítógépet tervezési segédeszközként. Ezért a pályázatok elbírálói úgy döntöttek, hogy a tervek egzaktabb összehasonlítása, valamint a nem szakmai közönség tájékoztatása érdekében, a terveket a számítógépes grafika eszközeivel jelenítik meg.

Így a második forduló valamennyi résztvevője — vezető külföldi CAD irodákkal együttműködve —, számítógéppel feldolgozott tervekkel pályázott. A Trias 3D CAD Studio a Gerő-Kévés-Szabó tervezőhármas által megálmodott terveket „gépesítette". A tervek megjelenítéséhez egy 256 színárnyalatot tudó, elektrosztatikus A0-s plottert használtak. Az eredmény nem is maradt el, második helyezést értek el az alkotók.

A világkiállítás kapcsán tehát a Trias 3D CAD szoftver városrendezési felhasználásával találkoztunk. Ám a rendszert építészeti, belsőépítészeti és látványtervezési feladatok megoldásához is jól alkalmazhatjuk.

Ennek a hazai fejlesztésű szoftvernek az az érdekessége, hogy a tervező szövegformátumú adatbázist hoz létre, és a program ennek tartalmát jeleníti meg grafikusan. A rendszerbe integráltak egy szövegszerkesztőt, egy grafikus interpretert, valamint egy háromdimenziós felületmodellel dolgo-



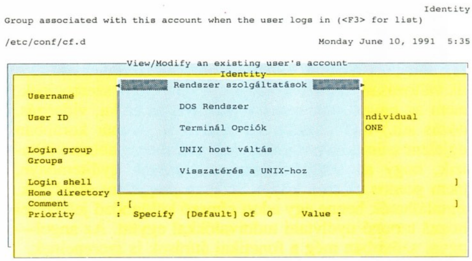
zó CAD rendszert. A rendszerre jellemző az is, hogy az adatbázis a felhasználó előtt mindig nyitott. Szintén kényelmes, hogy az adatbázist nem kell kézzel feltölteni – ezt egy piktogramos menüvel is megtehetjük. A rendszer anyag- és mennyiségi kigyűjtést is végez, amely költségvetési programhoz illeszthető.

Ezzel a CAD rendszerrel komplex feladatokat oldhatunk meg. Tudásának megfelelően a szoftver ára is igen magas, 450 000 Ft. S ha hozzávesszük azt az igényes hardvert (IBM AT 80386, 2 MB RAM, nagy felbontású grafikus monitor), amelyen ez a szoftver fut, akkor talán inkább eseti megbízások, kiviteli és látványtervek elkészítéséhez érdemesebb igénybe venni a Trias 3D CAD Studio szolgáltatásait.

UNIX? DOS?

Bizonyára sok felhasználó találta szembe magát azzal a problémával, hogy mit tegyen, ha már sok DOS-programja van, de a feladatok növekedése más operációs rendszert kíván. Egy másik operációs rendszerre való áttéréskor senki sem akarja elveszíteni megszokott alkalmazásait és adatait. Ugyanakkor általában kevés pénz áll rendelkezésre egy nagy teljesítményű gép és a köré épülő munkaállomások megvásárlására.

Ezen a gondon próbál segíteni a Xeus Rendszerépítő Iroda — KFKI által kifejlesztett Xeus nevű szoftver, amely egy UNIX-alapú intelligens terminálkezelő rendszer. Ezek a terminálok alkalmasak arra, hogy átvállaljanak egy-két feladatot a központi géptől, például a terminál I/O műveleteit vagy azt, hogy néhány programrészt a terminálon futtassunk. Sőt a DOS alatt működő terminálokról esetenként még a központi gép háttértárához is hozzáférhetünk. A Xeus rendszer tehát



egy UNIX-ot futtató központi gépből és az azt körülvevő IBM PC-kompatibilis, MS-DOS alatt működő terminálokból áll, ahol a gépek ARCNET vagy ETHERNET hálózaton keresztül kommunikálhatnak. A Xeus lehetővé teszi, hogy a terminálokon dolgozó felhasználók régi DOS-alkalmazásaikat futtassák az új operációs rendszer tanulása közben. Így a rendszer egyfajta sima áttérést tesz lehetővé a DOS-ról a UNIX-ra.

Ez a „hiánypótló", valóban okos szoftver méltán váltott ki nagy érdeklődést az Ifabón. A Xeus rendszer közel 80 000 Ft-os árával (10 terminálig) talán nem riasztja el az ilyen jellegű problémák megoldása előtt álló felhasználókat.

Számítógépes szótár

A könyvesboltokban a közeljövőben jelenik meg az Akadémiai Kiadó gondozásában az új angol—magyar, magyar—angol szótár. Az új szótár kiadásáról azért adunk hírt, mert

Kivágható postautalvány a PC Turbo Klub tagdíjának befizetéséhez. (Évi 2.112,– forint)

ÁTUTALÁSI POSTAUTALVÁNY

Ft f, azaz

Ft fillérről

A befizető
neve és címe

380–66760
számla javára

Bevételi szám:

Ellenőrző szám:

Keletbélyegző

Az összeg rendeltetése

PC Turbo Klub tagsági díj

ÉRTESÍTÉS

Ft f, azaz

Ft fillérről

A befizető
neve és címe

Jelölő adat

380–66760
számla javára

CÉDRUS
Informatikai Részvénytársaság

PC Turbo Klub tagsági díj

FELADÓVEVÉNY

Ft f, azaz

Ft fillérről

A befizető
neve és címe

380–66760
számla javára

CÉDRUS
Informatikai Részvénytársaság

Bevételi szám:

A felvevőhivatal keletbélyegzője

A felvevő aláírása

számítástechnikai feldolgozással készült. A szegedi Scriptum Kft. készítette el a szótár programváltozatát. A szótárpár 30, illetve 35 ezer címszót, és 12, illetve 20 ezer szókapcsolatot, kifejezést tartalmaz. A szótárban nemcsak mindennapi életünk változásait tükröző új szavak és fordulatok szerepelnek, hanem az egyes szakterületek (számítástechnika, videózás, autózás stb.) legfontosabb kifejezései is. A már korábban megjelent számítógépes szótáraktól elsősorban abban különbözik, hogy az adatbázis nemcsak szavak gyűjteménye, hanem pontos mása a könyv alakban megjelent szótárnak. Megtalálhatók benne egy adott címszó különböző jelentései a hozzá tartozó nyelvtani tudnivalókkal együtt. Az angol—magyar szótárban még a fonetikai átírások is szerepelnek.

Ez a szótárprogram valamennyi IBM-kompatibilis számítógépen, DOS 3.0 vagy magasabb verziószámú operációs rendszer alatt működik, feltéve, ha van 4 MB winchester-kapacitásunk. Bár a programot védelemmel látták el, 5000 Ft-os árával ennek ellenére vélhetően sokan hozzáférhetnek. A regisztrált felhasználók ezért természetesen a kiegészítéseket is megkapják a lemezár és a postaköltség fejében.

A Scriptum Kft. idén őszre tervezi, hogy az általa készített szótár rugalmasabban felhasználható legyen, tehát mi magunk is bővíthessük új szavak felvételével. A program lehetőséget nyújt arra, hogy a pontatlanul beírt szavakat is visszakeressük. A szótár részeként a fordítást segítő eljárás tulajdonképpen egy adott szöveg kiszótárazását végzi el. A szöveghez tartozó szavakat kikeresi és ezekből külön fájlt hoz létre. Ha egy szóhoz több jelentés is kapcsolódik, akkor az összes variációt felajánlja.

Az Akadémiai Kiadó és a Scriptum Kft. között az együttműködés annyira sikeresnek bizonyult, hogy egy szótársorozat közös kiadását is tervezik: hamarosan megjelentetik a magyar helyesírási szótárt, valamint a német, francia, spanyol, román szótárt is.

N/Joy – az irodák világa

Az IQ Soft által forgalmazott szoftver olyan összetett adminisztrációs rendszer, amely egyesíti a szövegszerkesztést, a dokumentumok tárolását és visszakeresését, valamint a táblázatkezelést és a hozzá tartozó diagramok elkészítését is.

Az N/Joy irodaautomatizálási rendszert a számítástechnikához kevéssé értő adminisztrátorok is könnyen megtanulják és megjegyzik. Ezért helyettesítik például rajzok a menüket. A felhasználó ugyanis jobban el tudja képzelni és meg tudja jegyezni a rajzokhoz tartozó jelentést, mint ami szöveges menüponthoz kötődik.

A program indításakor egy irodába lépünk, amely „szobákból" áll. Ezeket a szobákat a felhasználó rendezheti be annak megfelelően, hogy milyen jellegű munkát kíván majd végezni. A szobában katalógusok vannak, ezek objektumokból állnak, amelyek adat-, tároló- vagy funkcionális objektumok lehetnek. Ezek az objektumok közvetlenül kezelhetők, paraméterezhetők, és másolat készíthető róluk.

A programcsomag meglehetősen igényes hardvert feltételez (Intel 80 386-os IBM-kompatibilis számítógépet, 4 MB RAM-ot, 40 MB fix lemezterületet és OS/2 operációs rendszert). Úgy véljük, hogy akiknek már megvan ez a hardverháttér, azoknak érdemes a kb. 90.000 Ft-os programcsomag megvásárlásán gondolkodniuk. Élvezzék ők is az N/Joy-t!

Sziebig Andrea

Kivágható postautalvány a PC Turbo Klub tagdíjának befizetéséhez. (Évi 2.112,– forint)

Tud............. /19.............sz.

A feladó (meghatalmazottja) felszólalt:

............, 1991 .... hó ..... -ig

A feladónak az összeg rendeltetésére vonatkozó közleménye

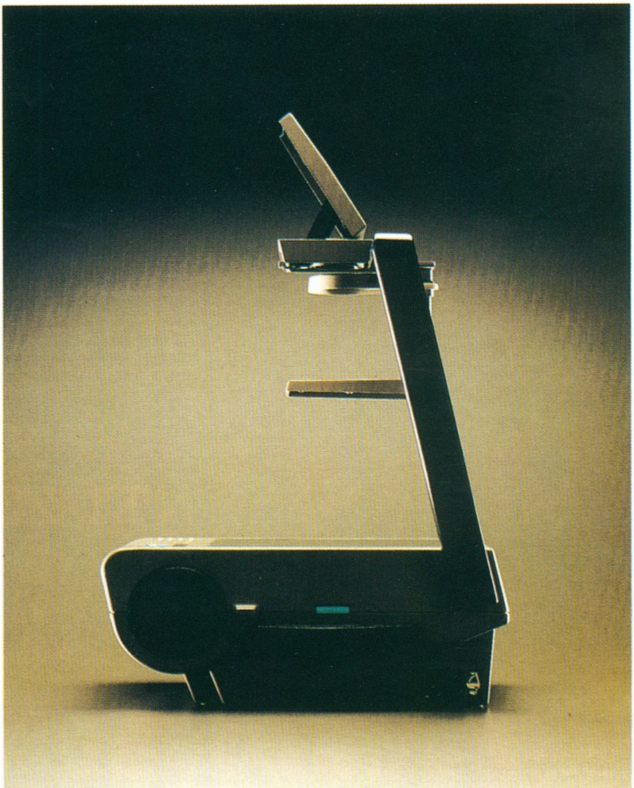
A bankszerv teljesítését igazoló bélyegzőnyomat:

Ez a helyes írásvetítő!

Nappali fény
mellett is
kitűnő képet
varázsolnak
a vászonra
a Polaroid
írásvetítők.

A könnyen
hordozható
legkisebb
modell
mindössze
4,5 kg.

Előadások
látványossá
tételéhez
ideális
útitárs!



Többféle változatban kaphatók a

FLOPPYLAND

számítástechnikai szaküzletben
(Budapest V., Váci utca 84. Telefon/Fax: 118-2651)
és országszerte a Cédrus Rt. viszonteladóinál.

Polaroid

INFORMÁCIÓKÉRÉS: 27 ▲

ALPHA MICROSYSTEMS
AMERIKAI CSÚCSTECHNOLÓGIA
MAGYARORSZÁGON



CREATIVE COMPUTER SOLUTIONS